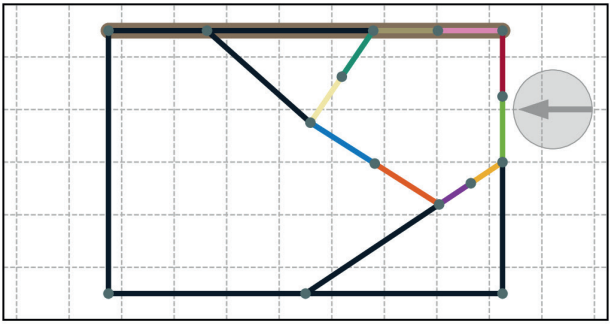
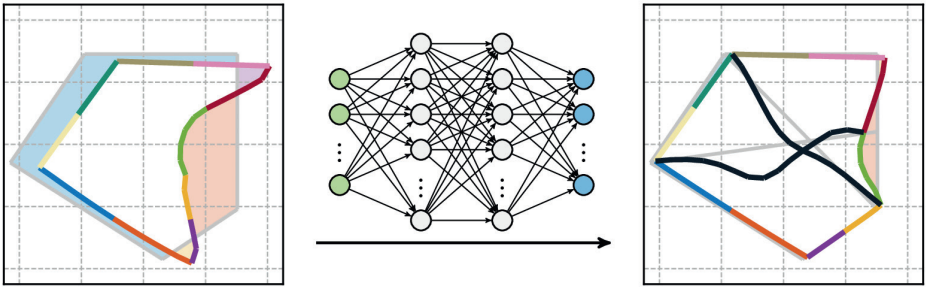
Unterstützung der Graphen- und Heuristikbasierten Topologieoptimierung crashbelasteter Strukturen durch Reinforcement Learning

Bergische Universität Wuppertal
Lehrstuhl für Optimierung mechanischer Strukturen

Jens Trilling





BERGISCHE
UNIVERSITÄT
WUPPERTAL

Unterstützung der Graphen- und Heuristikbasierten Topologieoptimierung crashbelasteter Strukturen durch Reinforcement Learning

Dissertation
zur Erlangung eines Doktorgrades

in der
Fakultät für Maschinenbau und Sicherheitstechnik
der
Bergischen Universität Wuppertal

vorgelegt von
Jens Trilling
aus Wuppertal

Wuppertal 2024

Tag der mündlichen Prüfung: 08.05.2024

Berichte aus dem Maschinenbau

Jens Trilling

Unterstützung der Graphen- und Heuristikbasierten Topologieoptimierung crashbelasteter Strukturen durch Reinforcement Learning

Shaker Verlag
Düren 2024

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: Wuppertal, Univ., Diss., 2024

Copyright Shaker Verlag 2024
Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN 978-3-8440-9554-8
ISSN 0945-0874

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren
Telefon: 02421 / 99 0 11 - 0 • Telefax: 02421 / 99 0 11 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

Jens Trilling

Unterstützung der Graphen- und Heuristikbasierten Topologieoptimierung crashbelasteter Strukturen durch Reinforcement Learning

Dissertation, Bergische Universität Wuppertal,
Fakultät für Maschinenbau und Sicherheitstechnik,
Lehrstuhl für Optimierung mechanischer Strukturen, Juni 2024

Kurzfassung

Die Auslegung crashrelevanter Strukturkomponenten ist eine zentrale Aufgabe in der Entwicklung von Fahrzeugen. Automatisierte Verfahren zur Optimierung solcher Strukturen sind rar. Rein mathematische Optimierungsmethoden scheitern an der Komplexität der Bestimmung von Ableitungen der relevanten Zielfunktionen und Restriktionen in Abhängigkeit von den Entwurfsvariablen. Aus diesem Grund wurde die *Graphen- und Heuristikbasierte Topologieoptimierung* (GHT) entwickelt, die den Optimierungsprozess mit aus Expertenwissen abgeleiteten Regeln, den Heuristiken, steuert. Innerhalb der GHT wird der Querschnitt von Extrusionsprofilen durch mathematische Graphen beschrieben, welche eine leichte Modifikation der Form und Topologie der Struktur durch die Heuristiken ermöglichen.

In dieser Dissertation wird eine neue Heuristik vorgestellt, welche ergänzend und konkurrierend zu den bestehenden Expertenregeln antritt. Durch Methoden des *Reinforcement Learnings* (RL) lernt das der Heuristik zugrundeliegende *Machine Learning*-Modell (ML-Modell), der Agent, selbstständig eine Strategie an, um die zu optimierende Struktur durch Topologieänderungen zu versteifen. Dazu wird eine zellenbasierte RL-Umgebung vorgestellt, welche eine konsistente Beschreibung von lokalen Bereichen in den Strukturgraphen ermöglicht. Die Zelle definiert den Bereich, in dem die Heuristik Topologiemodifikationen vornehmen kann. Durch ein neu entwickeltes Formabweichungsmaß wird die Steifigkeit einer Zelle rein geometrisch beschrieben. Das Training der Agenten basiert auf einem dreistufigen Prozess, bei dem in den ersten beiden Stufen systematisch nach geeigneten Trainingsparametern gesucht wird. In der dritten Stufe soll die Strategie des Agenten durch Transferlernen weiter generalisiert werden. Neben der versteifenden Heuristik wird diskutiert, wie und ob auf Basis des vorgestellten Ansatzes eine weitere RL-Heuristik entwickelt werden kann, welche die Strukturen nachgiebiger macht.

Die strukturversteifende Heuristik wird in verschiedenen praktischen GHT-Optimierungen auf ihre Performance und den Mehrwert für die GHT hin untersucht. Analysiert wird ein Rahmenmodell, ein Biegeträger und ein Schwellerausschnitt in unterschiedlichen Crashlastfällen. In vielen Optimierungen hat sich gezeigt, dass die RL-Heuristik erfolgreich eingesetzt werden kann und den Optimierungsprozess zu einer besseren Struktur gegenüber einer Vergleichsoptimierung ohne RL-Heuristik führt.

Stichworte: Strukturoptimierung, Topologieoptimierung, Crashoptimierung, Künstliche Intelligenz, Reinforcement Learning

Jens Trilling

Support for the Graph- and Heuristic based Topology optimization of crash-loaded structures through Reinforcement Learning

PhD thesis, University of Wuppertal,
School of Mechanical Engineering and Safety Engineering,
Chair for Optimization of Mechanical Structures, June 2024

Abstract

The development of vehicles relies greatly on the design of crash-relevant structural components. However, the optimization of such structures through automated methods is difficult. Purely mathematical optimization methods fail due to the complexity of determining the derivatives of the relevant objective functions and constraints as a function of the design variables. Therefore, the *Graph and Heuristic based Topology optimization* (GHT) was developed, which guides the optimization process with rules derived from expert knowledge, known as heuristics. The cross-section of extrusion profiles is described by mathematical graphs within the GHT, allowing the shape and topology of the structure to be easily modified by heuristics.

In this dissertation, a novel heuristic is presented that both complements and challenges established expert rules. Using *Reinforcement Learning* (RL) techniques, the *Machine Learning* (ML) model underlying the heuristic, called the agent, autonomously learns a strategy to stiffen the structure to be optimized through topological modifications. To ensure a consistent description of local domains within the structure graphs, a cell-based RL environment is introduced. The cell delineates the region within which the heuristic can apply topological modifications. A newly developed shape deviation measure describes the stiffness of a cell geometrically. The agents are trained via a three-stage process, wherein the initial two stages entail a methodical search for appropriate training parameters. Subsequently, in the third stage, the agent's strategy is further generalized through transfer learning. Beside the stiffening heuristic, the possibility of developing another RL heuristic based on the presented approach, which makes the structures more compliant, is discussed.

The structure stiffening heuristic is investigated in various practical GHT optimizations for its performance and added value to the GHT. Analyses are carried out on a frame model, a bending beam and a rocker cutout in various crash load cases. In many optimizations it is shown that the RL heuristic can be successfully applied and leads the optimization process to a better structure compared to a benchmark optimization without the RL heuristic.

Keywords: Structural Optimization, Topology Optimization, Crash Optimization, Artificial Intelligence, Reinforcement Learning

Danksagung

Die vorliegende Dissertation ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl *Optimierung mechanischer Strukturen* der *Bergischen Universität Wuppertal* entstanden. Hier möchte ich mich bei allen Personen bedanken, welche einen wertvollen Beitrag zur erfolgreichen Vollendung der Dissertation geleistet haben.

Zunächst möchte ich mich bei meinen Eltern *Gabriele & Ulrich Trilling* bedanken, welche mich stets in meinen Zielen tatkräftig unterstützt haben und mir den Weg dorthin geebnet haben.

Ebenso möchte ich mich bei meinem geschätzten Freund *Yannis Rohloff* bedanken, welcher stets ein offenes Ohr hatte und als Informatiker fundierte Ideen zur Verbesserung meiner Forschungsergebnisse beisteuern konnte.

Weiteren Dank möchte ich *Georgios Ioannou* widmen, welcher als studentische und wissenschaftliche Hilfskraft mit großem Interesse am Themengebiet der künstlichen Intelligenz wertvolle Beiträge und Diskussionen ermöglicht hat, die mich in meiner Arbeit weitergebracht haben. Durch seine von mir betreuten Arbeiten konnte ich den inhaltlichen Fokus dieser Dissertation weiter schärfen.

Meine *Kollegen* hatten in Fachgesprächen stets geschätzte Hinweise, wie und an welchen Stellschrauben ich die von mir entwickelte Methodik weiter verbessern kann. Für den aufgebrachten Einsatz möchte ich mich bei allen Kollegen herzlich bedanken. Hervorheben möchte ich an dieser Stelle *Dr. Dominik Schneider*, welcher sich in vielen Belangen gesondert Zeit genommen hat. Das gilt für projektbezogenen Treffen mitsamt seinen äußerst hilfreichen Vorschlägen und insbesondere für die große Unterstützung in programmiertechnischen Belangen seitens der Graphen- und Heuristikbasierten Topologieoptimierung.

Ganz besonderer Dank gebührt meinem Doktorvater *Prof. Dr. Axel Schumacher*, welcher mir die Dissertation ermöglicht hat und sie mit seinen inhaltlichen Anregungen aus zahlreichen Fachgesprächen aufgewertet hat.

Der größte Dank geht an meine Partnerin *Maria Huber*, die sich auf lange und inhaltlich intensive Gespräche eingelassen hat und meiner Forschung und Dissertation trotz Fachfremdheit unschätzbar wertvollen Mehrwert geben konnte.

Wuppertal im Juni 2024 Jens Trilling

Inhaltsverzeichnis

Abkürzungs- und Symbolverzeichnis **V**

1 Einleitung **1**
1.1 Problemstellung und Motivation 1
1.2 Zielsetzung und Eingrenzung 2
1.3 Aufbau der Dissertation 3

2 Grundlagen der Graphen- und Heuristikbasierten Topologieoptimierung **5**
2.1 Anforderungen an Crashstrukturen 5
2.2 Finite-Elemente-Methode im Kontext von Crashsimulationen 7
2.3 Grundlagen der automatisierten Strukturoptimierung 9
2.4 Methoden zur Topologieoptimierung crashbelasteter Strukturen 13
2.5 Graphen- und Heuristikbasierte Topologieoptimierung 16
2.5.1 Geometriebeschreibung durch Graphen 16
2.5.2 Aufbau des Simulationsmodells 18
2.5.3 Fertigungsrestriktionen 19
2.5.4 Vorstellung der Heuristiken 20
2.5.5 Optimierungsablauf 23

3 Grundlagen des Machine Learnings mit Fokus auf Reinforcement Learning **27**
3.1 Einführung und Einordnung Künstlicher Intelligenz 27
3.2 Grundlagen des Machine Learnings 29
3.2.1 Strukturierte und unstrukturierte Daten 29
3.2.2 Lernparadigmen des Machine Learnings 30
3.2.3 Feature-Engineering 34
3.2.4 Modellauswahl 35
3.3 Neuronale Netze 38
3.3.1 Multilayer Perceptron 38
3.3.2 Convolutional Neural Network 42
3.4 Reinforcement Learning 44
3.4.1 Konzeptionelle Einführung in das Reinforcement Learning 45
3.4.2 Theoretische Grundlagen 46
3.4.3 Grundlegende Lernverfahren 49
3.4.4 Approximationsverfahren 52

4 Stand der Technik zur Entwicklung von Crashstrukturen gestützt durch Machine Learning 57
4.1 Aufbereitung von Simulationsdaten . . . 57
4.2 Ingenieurtechnische Problemstellungen . . . 59
4.2.1 Vorhersage von Strukturantworten . . . 59
4.2.2 Identifikation von Deformationsmodi . . . 60
4.2.3 Strukturoptimierung . . . 62
4.2.4 Streuanalyse . . . 64

5 Heuristikentwicklung mit Reinforcement Learning zur Strukturversteifung 65
5.1 Möglichkeiten des Machine Learnings zur Unterstützung der Graphen- und Heuristikbasierten Topologieoptimierung . . . 65
5.2 Anforderungen an die Heuristik . . . 69
5.3 Funktionsprinzip der Heuristik . . . 69
5.4 Wahl des Simulationsmodells . . . 72
5.5 Implementierung der Trainingsumgebung . . . 73
5.5.1 Aufbau der Umgebung . . . 73
5.5.2 Graphbasierte Datenstrukturen . . . 77
5.5.3 Generierung von Zufallsgraphen . . . 81
5.5.4 Generierung von Zufallslastfällen . . . 83
5.5.5 Identifikation valider Zellen . . . 86
5.5.6 Aufbau und Simulation des Modells . . . 88
5.5.7 Evaluationsmaß für Strukturperformance . . . 89
5.5.8 Reward-Funktion . . . 91
5.5.9 Aktionsraum . . . 92
5.5.10 Durchführung der Topologieänderungen . . . 93
5.5.11 Observationsraum . . . 94
5.6 Aufbau und Funktionsweise der Agenten . . . 109
5.6.1 Auswahl und Beschreibung des Trainingsalgorithmus . . . 109
5.6.2 Trainingsschleife . . . 111
5.6.3 Standardisieren der Observationen . . . 114
5.6.4 Konzeptionelle Architektur der Agenten . . . 115
5.7 Training der Agenten . . . 118
5.7.1 Vorgehen . . . 118
5.7.2 Datensatzgenerierung . . . 119
5.7.3 Datensatzanalyse . . . 121
5.7.4 Aufbau der datensatzbasierten Umgebung . . . 122
5.7.5 Wahl relevanter Observationsfeatures . . . 123
5.7.6 Hyperparameterstudie . . . 132
5.7.7 Transferlernen und Modellauswahl . . . 135
5.7.8 Anschauliche Bewertung der trainierten Agenten . . . 138

5.8 Fortführende Diskussion des Trainingsverhaltens 141
5.9 Integration der Heuristik in den Optimierungsprozess 143

6 Adaption der entwickelten Heuristik zur Generierung weicher Strukturen 145
6.1 Modifikation des Optimierungsziels des Agenten 145
6.2 Entfernen von Wänden aus Zellen 149
6.3 Einsatz von etablierten Dimensionierungsstrategien 151

7 Einsatz der entwickelten Heuristik in praktischen Optimierungen 153
7.1 Rahmenmodell 154
7.1.1 Modellvorstellung 154
7.1.2 Minimierung der maximalen Impaktorverschiebung 155
7.1.3 Minimierung der maximalen Impaktorbeschleunigung 168
7.2 Biegeträger 172
7.2.1 Modellvorstellung 172
7.2.2 Minimierung der maximalen Impaktorverschiebung 173
7.3 Schwellerausschnitt 178
7.3.1 Modellvorstellung 178
7.3.2 Minimierung der Pfahlintrusion 179
7.4 Diskussion der Heuristikperformance 184

8 Zusammenfassung 187

9 Ausblick 191

Vorveröffentlichungen zu dieser Dissertation 193

Literaturverzeichnis 194

Anhang

A Trainingshistorien der Hyperparameteruntersuchung **205**

B Tabellarische Zusammenfassungen der praktischen Optimierungen **207**

B.1 Rahmenmodell 207

B.1.1 Minimierung der Impaktorverschiebung 207

B.1.2 Minimierung der maximalen Impaktorbeschleunigung 211

B.2 Biegeträger 212

B.3 Schwellerausschnitt 213

Abkürzungs- und Symbolverzeichnis

Abkürzungen

ABS	Antiblockiersystem
Adam	Adaptive Moment Estimation
ASCII	American Standard Code for Information Interchange
BED	Balance Energy Density
CAE	Computer Aided Engineering
CFC	Channel Frequency Class
CNN	Convolutional Neural Network
DiESL	Difference-based Equivalent Static Loads
DNW	Delete Needless Walls
DoE	Design of Experiments
DoF	Degrees of Freedom
DT	Decision Tree
EA-LSM	Evolutionary Level-Set-Method
EGO	Efficient Global Optimization
ESLM	Equivalent Static Loads Method
ESP	Elektronisches Stabilitätsprogramm
FE	Finite-Elemente
FEM	Finite-Elemente-Methode
GAE	Generalized Advantage Estimate
GCN	Graph Convolutional Network
GHT	Graphen- und Heuristikbasierte Topologieoptimierung
GN	Geometrische Normierung
GNN	Graph Neural Network
GRAMB	Graph and Mechanics Builder
GSA	Ground Structure Approach
HCA	Hybrid Cellular Automaton
HCA-TWS	Hybrid Cellular Automaton for Thin-Walled Structures

HIC	Head Injury Criterion
HPC	High-Performance Computing
ID	Identifikationsnummer
KI	Künstliche Intelligenz
kNN	k-Nearest Neighbors
LFR	Lockere Fertigungsrestriktionen
LHS	Latin Hypercube Sampling
LSTM	Long Short-Term Memory
MEP	Markow-Entscheidungsproblem
ML	Machine Learning
MLP	Multilayer Perceptron
MNIST	Modified National Institute of Standards and Technology
MSE	Mean Squared Error
NEAT	NeuroEvolution of Augmented Topologies
NN	Neuronales Netz
OBB	Oriented Bounding Box
PCA	Principal Component Analysis
PFI	Permutation Feature Importance
PFR	Praxisnahe Fertigungsrestriktionen
PPO	Proximal Policy Optimization
ReLU	Rectified Linear Unit
RF	Random Forest
RL	Reinforcement Learning
RLS	Reinforcement Learning: Stiffness
SBW	Support Buckling Walls
SL	Supervised Learning
SLE	Split Long Edges
SRSM	Successive Response-Surface-Model
SSO	Size and Shape Optimization
SVM	Support Vector Machine
Tanh	Tangens Hyperbolicus
TD	Temporal Difference
t-SNE	t-distributed Stochastic Neighbor Embedding
UDSC	Use Deformation Space Compression
UDST	Use Deformation Space Tension
UL	Unsupervised Learning

Mathematische Notationen und Operatoren

Allgemein

Symbol	Bedeutung
x	Skalar
$\boldsymbol{x}$	Spaltenvektor
$\boldsymbol{x}^\intercal$	Zeilenvektor
x_i	i-te Komponente des Vektors $\boldsymbol{x}$
$\left\|\boldsymbol{x}\right\|_p$	p-Norm des Vektors $\boldsymbol{x}$, $\left\|\boldsymbol{x}\right\|_p = \left(\sum_{i=1}^{n} \left|x_i\right|^p\right)^{1/p}$
$\boldsymbol{X}$	Matrix
$\boldsymbol{X}^{-1}$	Inverse einer Matrix
$x \times y \times z$	Dimension eines n-dimensionalen Arrays, hier mit $n = 3$
$\tilde{x}$	Normierter Wert *oder* modellbasierte Vorhersage
$\overline{x}$	Mittelwert
$\hat{x}$	Schätzwert *oder* Achse in lokalem Koordinatensystem
x_*	Wert im Optimum
Δx	Differenz
$x \leftarrow y$	Zuweisung
$x \lesssim y$	x ist in etwa kleiner als y
$f\left(x\right)$	Funktion f in Abhängigkeit von x
$f'\left(x\right)$	Erste Ableitung einer Funktion f nach x
$\dot{x}$, $\ddot{x}$	Erste und zweite Ableitung von x nach der Zeit
$\frac{\partial x}{\partial y}$	Partielle Ableitung von x nach y
$\nabla f(\boldsymbol{x})$	Gradient von Funktion f im Punkt $\boldsymbol{x}$
$f * g$	Faltung der Funktionen f und g
$f(x)\Big\|_y$	Evaluation von $f(x)$ an der Stelle y
n_{x}	Anzahl einer Größe x
$\mathrm{clip}\left(x; a; b\right)$	Limitieren von x außerhalb von $[\,a;\,b\,]$ $(= \min\left(\max\left(x, a\right), b\right))$
# Größe	Anzahl einer Größe

Mengenlehre und Aussagenlogik

Symbol	**Bedeutung**
$\mathcal{X}$	Menge
$\{x; y; z\}$	Menge der Elemente x, y und z
$\emptyset$	Leere Menge
$\mathbb{N}$	Menge der natürlichen Zahlen
$\mathbb{N}_0$	Menge der natürlichen Zahlen inklusive der Null
$x \in \mathcal{X}$	Element x aus der Menge $\mathcal{X}$
$\forall x \in \mathcal{X}$	Für alle Elemente x in der Menge $\mathcal{X}$
$\partial\mathcal{X}$	Rand der Menge $\mathcal{X}$
$\lvert\mathcal{X}\rvert$	Mächtigkeit der Menge $\mathcal{X}$ (Elementanzahl)
$\mathcal{X} \setminus \mathcal{Y}$	Differenzmenge von $\mathcal{X}$ und $\mathcal{Y}$ ($\mathcal{X}$ ohne $\mathcal{Y}$)
$\mathcal{X} \cap \mathcal{Y}$	Konjunktion der Mengen $\mathcal{X}$ und $\mathcal{Y}$ (Schnittmenge)
$x \wedge y$	Konjunktion der logischen Ausdrücke x und y (Und-Verknüpfung)
$[\,x;\, y\,]$	Geschlossenes Intervall von x nach y

Wahrscheinlichkeitsrechnung und Kombinatorik

Symbol	**Bedeutung**
$\Pr(X = x)$	Wahrscheinlichkeit, dass die Zufallsvariable X den Wert x annimmt
$\Pr(X = x \mid y)$	Bedingte Wahrscheinlichkeit, dass die Zufallsvariable X den Wert x bei gegebenem Wert y annimmt
$x \sim \mathcal{N}(\mu, \sigma^2)$	Sample x aus einer Normalverteilung mit Erwartungswert μ und Standardabweichung σ
$\mathbb{E}[X]$	Erwartungswert der Zufallsvariable X
$\binom{n}{k}$	Binomialkoeffizient

Mathematische Graphen im Kontext der Graphen- und Heuristikbasierten Topologieoptimierung

Symbol	Bedeutung
$x_{e_i}^{\{G\}}$	Kantenspezifischer Wert x von Kante e_i eines Graphen G
$x_{e_i,e_j}^{\{G\}}$	Wert x zwischen Kanten e_i und e_j eines Graphen G
$< x >$	Zum Index x im Adjazenzvektor des modifizierten Zellengraphen G_{Zm} zugehörige Kante

Zeichen

Allgemein

Zeichen	Bedeutung
i, j, k, l, m	Zählvariablen
t	Zeit bzw. Zeitpunkt
t_0	Ausgangszeit bzw. -zeitpunkt
x, y, z	Raumrichtungen im globalen Koordinatensystem
$\hat{x}$, $\hat{y}$, $\hat{z}$	Raumrichtungen in lokalem Profilkoordinatensystem

Wahrscheinlichkeitsrechnung

Zeichen	Bedeutung
p	Wahrscheinlichkeit für ein Ereignis, wie z. B. $p = \Pr(X = x)$
$\overline{x}$	Empirischer Mittelwert (Mittel einer Stichprobe)
s_{emp}	Empirische Standardabweichung
μ	Erwartungswert
σ	Standardabweichung
$\mathcal{N}(\mu, \sigma^2)$	Normalverteilung

Mechanik

Zeichen	Bedeutung
$a(t)$	Beschleunigungsverlauf über die Zeit t
$E_{\mathrm{i},k}$	Innere Energie einer Strukturkomponente k
m_{init}	Initiale Masse
$\boldsymbol{D}$	Geschwindigkeitsproportionale Dämpfungsmatrix
$\boldsymbol{f}$	Aufgeprägter Lastvektor
$\boldsymbol{K}$	Steifigkeitsmatrix
$\boldsymbol{M}$	Massenmatrix
$\boldsymbol{u}$	Verschiebungsvektor
$\dot{\boldsymbol{u}}$	Geschwindigkeitsvektor
$\ddot{\boldsymbol{u}}$	Beschleunigungsvektor

Optimierung

Zeichen	Bedeutung
$f(\boldsymbol{x})$	Zielfunktion
$g_j(\boldsymbol{x})$	Ungleichheitsrestriktionen
$h_k(\boldsymbol{x})$	Gleichheitsrestriktionen
n_g	Anzahl Ungleichheitsrestriktionen
n_h	Anzahl Gleichheitsrestriktionen
$\boldsymbol{x}$	Entwurfsvariablen
$x_i^{(l)}$, $x_i^{(u)}$	Untere und obere Entwurfsvariablengrenzen
$\Phi(\boldsymbol{x})$	Level-Set-Funktion
$u_{y,\max}$	Maximale Verschiebung in y-Richtung ($u_{y,\max} := \max\, u_y(t)$)
$u_{-y,\max}$	Maximale Verschiebung in negative y-Richtung ($u_{-y,\max} := \max\, -u_y(t)$)

Zeichen	Bedeutung
$\ddot{u}_{-y,\max}$	Maximale Beschleunigung in negative y-Richtung ($\ddot{u}_{-y,\max} := \max -\ddot{u}_y(t)$)
$u_{-z,\max}$	Maximale Verschiebung in negative z-Richtung ($u_{-z,\max} := \max -u_z(t)$)

Mathematische Graphen im Kontext der Graphen- und Heuristikbasierten Topologieoptimierung

Zeichen	Bedeutung
G_S	Strukturgraph
G_E	Evaluationsgraph
G_Z	Zellengraph
G_Zm	Modifizierter Zellengraph
e_i	Kante e_i eines Graphen
$A_{e_i}^{\{G\}}$	Querschnittsfläche der zu Kante e_i zugehörigen Wand eines Graphen G
$l_{e_i}^{\{G\}}$	Länge der Kante e_i in einem Graphen G
$d_{e_i,e_j}^{\{G\}}$	Abstand der Kanten e_i und e_j in einem Graphen G
$\alpha_{e_i,e_j}^{\{G\}}$	Winkel zwischen zwei Kanten e_i und e_j in einem Strukturgraphen
l_min	Mindestkantenlänge zum Erfüllen der Herstellrestriktionen
d_min	Mindestkantenabstand zum Erfüllen der Herstellrestriktionen
α_min	Mindestwinkel zwischen zwei Kanten zum Erfüllen der Herstellrestriktionen
t_min	Minimale Wandstärke zum Erfüllen der Herstellrestriktionen
t_max	Maximale Wandstärke zum Erfüllen der Herstellrestriktionen
t_eval	Auswertezeitpunkt für die Heuristiken
$\mathcal{V}^{\{G\}}$	Menge aller Vertices $\{v_i; v_j; \dots\}$ eines Graphen G
$\partial\mathcal{V}^{\{G\}}$	Menge aller Vertices, die auf dem Rand der konvexen Hülle der Vertex-Koordinaten eines Graphen G liegen
$\mathcal{V}^{\{e_i\}}$	Menge aller Vertices $\{v_j; v_k\}$, die die Kante e_i aufspannen

Zeichen	**Bedeutung**
v_i	Vertex v_i eines Graphen
$\mathcal{E}^{\{G\}}$	Menge aller Kanten $\{e_i; e_j; \dots\}$ eines Graphen G
$\partial\mathcal{E}^{\{G\}}$	Menge aller im Graphen G vorhandenen Kanten, die durch $\partial\mathcal{V}^{\{G\}}$ aufgespannt werden

Machine Learning und Neuronale Netze

Zeichen	**Bedeutung**
b_j	Bias eines j-ten Neurons
D	Datensatz $D = \{X, Y\}$
f_i	Feature i
L	Fehlerfunktion
n_p	Anzahl Datenpunkte
o_j	Output bzw. Aktivierung des j-ten Neurons
R^2	Bestimmtheitsmaß
w_{ij}	Gewicht von Input x_i zum j-ten Neuron
x	Feature bzw. Input eines Neurons
X	Features des Datensatzes D
y	Funktionswert
y_d	Wahrer Wert aus Datensatz bzw. Label
y_p	Vorhergesagter Wert durch Modell
Y	Labels des Datensatzes D
z_j	Ergebnis der Propagierungsfunktion eines j-ten Neurons
$\boldsymbol{I}$	Input-Array
$\boldsymbol{K}$	Kernel bzw. Filter
$\boldsymbol{S}$	Feature-Map
$\boldsymbol{z}$	Logits (Ergebnis der Propagierungsfunktion für den Output-Layer)
η	Lernrate

Zeichen	Bedeutung
Σ	Propagierungsfunktion
φ	Aktivierungsfunktion

Reinforcement Learning

Zeichen	Bedeutung
a	Konkrete Aktion $a \in \mathcal{A}$
A_t	Zufallsvariable zur Beschreibung der Aktion zum Zeitpunkt $t \in \mathbb{N}_0$
$A(s, a)$	Advantage-Funktion $A(s, a) = Q(s, a) - V(s)$
c_1	Zustandswertkoeffizient für $L_t^{\mathrm{VF}}(\boldsymbol{\theta})$
c_2	Entropiekoeffizient für $S\left[\pi_{\boldsymbol{\theta}}\right](s_t)$
G_t	Zufallsvariable des (un-)diskontierten Gewinns bzw. Returns
$J(\boldsymbol{\theta})$	Metrik zur Optimierung der Strategie
$\overline{L_{\mathrm{Ep}}}$	Mittlere Episodenlänge
$L_t^{\mathrm{CLIP}}(\boldsymbol{\theta})$	Clip-Term in Ersatzzielfunktion $L_t^{\mathrm{CLIP+VF+S}}(\boldsymbol{\theta})$
$L_t^{\mathrm{CLIP+VF+S}}(\boldsymbol{\theta})$	Ersatzzielfunktion zur Optimierung der Strategie in Abhängigkeit der Strategieparameter $\boldsymbol{\theta}$
$L_t^{\mathrm{VF}}(\boldsymbol{\theta})$	Zustandswertterm in Ersatzzielfunktion $L_t^{\mathrm{CLIP+VF+S}}(\boldsymbol{\theta})$
n_{batches}	Batchgröße für das Strategieupdate
n_{epochs}	Anzahl Epochen für das Strategieupdate
n_{steps}	Größe des Rollout-Buffers für das Strategieupdate
$S\left[\pi_{\boldsymbol{\theta}}\right](s_t)$	Entropieterm in Ersatzzielfunktion $L_t^{\mathrm{CLIP+VF+S}}(\boldsymbol{\theta})$
Q	Approximation von $q_\pi(s, a)$
$q_\pi(s, a)$	Aktionswertfunktion
$q_*(s, a)$	Maximierte Aktionswertfunktion $\forall s \in \mathcal{S}$ und $\forall a \in \mathcal{A}$
r	Konkrete Belohnung bzw. Reward $r \in \mathcal{R}$
R_t	Zufallsvariable zur Beschreibung des Rewards zum Zeitpunkt $t \in \mathbb{N}_0$
s	Konkreter Zustand $s \in \mathcal{S}$

Zeichen	**Bedeutung**
S_t	Zufallsvariable zur Beschreibung des Zustands zum Zeitpunkt $t \in \mathbb{N}_0$
s'	Folgezustand $s' \in \mathcal{S}$ aus Zustand s mit Aktion a
T	Terminaler Zeitpunkt $T \in \mathbb{N}$
V	Approximation von $v_\pi(s)$
V_t^{targ}	Wahre Zustandswerte
$v_\pi(s)$	Zustandswertfunktion
$v_*(s)$	Maximierte Zustandswertfunktion $\forall s \in \mathcal{S}$
$\hat{v}(s, \boldsymbol{w})$	Parametrisierte Zustandswertfunktion
$\boldsymbol{w}$	Parametervektor
α	Schrittweite
δ_t	Temporal-Difference-Error
ϵ	Clipping der Strategieänderung
γ	Diskontierungsfaktor
λ	Varianz-Bias-Trade-Off
$\pi(s)$	Deterministische Strategie
$\pi\left(a \mid s\right)$	Probabilistische Strategie
π_*	Optimale Strategie
π'	Neue Strategie
$\boldsymbol{\theta}$	Strategieparameter
$\xi_t(\boldsymbol{\theta})$	Verhältnis aus Wahrscheinlichkeitsverteilungen probabilistischer Strategien
$\mathcal{A}$	Menge an Aktionen
$\mathcal{P}$	Menge an Übergangswahrscheinlichkeiten
$\mathcal{R}$	Menge an reellwertigen Rewards für Zustandswechsel
$\mathcal{S}$	Menge an Zuständen

Umgebungsimplementierung

Zeichen	Bedeutung
$A\big\vert_t$	Aufgespannte Fläche einer Zelle ausgewertet zum Zeitpunkt t
$\tilde{A}_\Delta$	Formabweichungsmaß
$A_{\Delta,j}\big\vert_t$	Differenzfläche ausgewertet zum Zeitpunkt t
n_{Zs}	Anzahl an Seiten der Zelle G_{Zm} ($n_{\mathrm{Zs}} = \lvert\mathcal{S}^{\{G_{\mathrm{Zm}}\}}\rvert$)
p_{ins}	Bestrafung zum Einbringen von Kanten in Zelle
$r_{t+1}^{(\star)}$	Reward ohne Bestrafungsterm p_{ins}
$g_t^{(\star)}$	Return ohne Bestrafungsterm p_{ins}
$\overline{g_0}_{\gamma=1}$	Undiskontierter mittlerer Return ausgehend vom initialen Zustand
$\overline{g_0}_{\gamma=1,\max}$	Theoretisch maximal erreichbarer Wert für $\overline{g_0}_{\gamma=1}$
$\overline{g_0}^{(\star)}_{\gamma=1,\max}$	Variante von $\overline{g_0}_{\gamma=1,\max}$ ohne Bestrafungsterm p_{ins}
$t_{\max}^{\mathrm{F},i}$	Zeitpunkt, an dem die gefilterte Schnittkraft einer Wand i maximal ist
$t_{\max}^{\mathrm{IE}}$	Zeitpunkt, an dem die innere Energie der Zelle maximal ist
v_0	Initiale Geschwindigkeit
v_0'	Gesampelte, initiale Geschwindigkeit
$v_{\min}$	Minimale Geschwindigkeit des Impaktors
$v_{\max}$	Maximale Geschwindigkeit des Impaktors
V_k	Volumen einer Strukturkomponente k
$\boldsymbol{R}_{\hat{z}}(\beta_t)$	Rotationsmatrix um die Extrusionsachse $\hat{z}$ des Profils zur Eliminierung von Starrkörperrotation
$\hat{\boldsymbol{x}}$	Koordinaten der Vertices in lokalem Profilkoordinatensystem
$\hat{x}_{\mathrm{norm}}$	Normierte Profilkoordinate in $\hat{x}$-Richtung
$\hat{y}_{\mathrm{norm}}$	Normierte Profilkoordinate in $\hat{y}$-Richtung
β_t	Winkel zwischen der initialen Zellkonfiguration und der Zellkonfiguration zu Zeitpunkt t
$\delta_{t,\mathrm{rel}}$	Relative Verbesserung der Formabweichung zur Bemessung der Steifigkeitserhöhung einer Zelle
κ	Geschätzter Einfluss einer Zelle auf das Strukturverhalten

Zeichen	**Bedeutung**
μ_{v_0}	Erwartungswert der initialen Geschwindigkeit
σ_{v_0}	Standardabweichung der initialen Geschwindigkeit
$\mathcal{V}^{\{G_{\mathrm{Zm}}\}}$	Menge an Vertices entlang des Zellrahmens nach dem Kantenteilungsprozess einer Zelle G_{Zm}
$\mathcal{I}^{\{G_{\mathrm{Zm}}\}}$	Menge an einziehbaren Kanten in eine leere Zelle G_{Zm}
$\mathcal{I}_{\mathrm{init}}^{\{G_{\mathrm{Zm}}\}}$	Menge an initialen Kanten innerhalb einer Zelle G_{Zm}
$\mathcal{I}_{\mathrm{max}}^{\{G_{\mathrm{Zm}}\}}$	Menge an vom Agenten maximal einziehbaren Kanten in eine Zelle G_{Zm}
$\mathcal{I}_{\mathrm{komb}}^{\{G_{\mathrm{Zm}}\}}$	Menge möglicher Kombinationen einziehbarer Kanten einer Zelle G_{Zm} für ein festgelegtes $\lvert\mathcal{I}_{\mathrm{max}}^{\{G_{\mathrm{Zm}}\}}\rvert$
$\mathcal{E}_{\mathrm{ges}}^{\{G_{\mathrm{Zm}}\}}$	Menge aller Kanten der Zelle G_{Zm}
$\partial\mathcal{E}^{\{G_{\mathrm{Zm}}\}}$	Menge der Kanten auf dem Rahmen einer Zelle G_{Zm}
$\mathcal{S}^{\{G_{\mathrm{Zm}}\}}$	Menge der Seiten der Zelle G_{Zm}

1 Einleitung

1.1 Problemstellung und Motivation

Die Sicherheit von Fahrzeuginsassen und beteiligten Verkehrsteilnehmer*innen hat höchste Priorität in der Entwicklung von crashrelevanten Strukturkomponenten von Fahrzeugen. Aus diesem Grund werden im Entwicklungsprozess mit *Computer Aided Engineering*-Methoden (CAE-Methoden) detaillierte Simulationsmodelle basierend auf der *Finiten-Elemente-Methode* (FEM) aufgebaut, mit denen das Crashverhalten prognostiziert werden kann. Die Auslegung einer Struktur, also die Verfolgung von Zielen und die Einhaltung von Restriktionen, ist aufgrund der komplexen Crashmechanik und des hohen Zeit- und Ressourcenaufwands von Crashsimulationen äußerst schwierig. Häufig werden bewährte Strukturkonzepte älterer Fahrzeugmodelle in die aktuelle Entwicklung übernommen und händisch auf die vorliegende Situation angepasst, da rein mathematische Optimierungsalgorithmen aufgrund der starken Nichtlinearität der Crashmechanik an ihre Grenzen stoßen.

Die *Graphen- und Heuristikbasierte Topologieoptimierung* (GHT) (Olschinka und Schumacher 2008; Ortmann und Schumacher 2013; Ortmann 2015; Ortmann et al. 2021) ist ein automatisiertes und in der Praxis etabliertes Verfahren zur Optimierung crashbelasteter Strukturen, was durch einen primär heuristischen Ansatz ermöglicht wird. Dabei treten verschiedene von Heuristiken vorgeschlagene Entwürfe in der aktuellen Iteration gegeneinander an, von denen die besten Entwürfe in die nächste Iteration übergeben werden. Zunächst wurde die GHT für die Optimierung von Extrusionsprofilen aus Aluminium entwickelt, sodass die optimierten Entwürfe dann durch Strangpressen gefertigt werden können. Beschrieben werden die Entwürfe durch einen mathematischen Graphen, der den Querschnitt des Extrusionsprofils beschreibt.

In den letzten Jahren wurde die GHT stets mit neuen Konzepten erweitert, um beispielsweise unterschiedliche Fertigungsverfahren in die Optimierung integrieren zu können (Link et al. 2018; Schneider et al. 2019) oder auch die drei-dimensionale Anordnung von Profilen in Rahmenstrukturen optimieren zu können (Beyer et al. 2021). Diese Dissertation soll ebenfalls einen Beitrag zur Weiterentwicklung der GHT liefern. Der Fokus liegt dabei auf Methoden des *Reinforcement Learnings* (RL), einem Teilbereich des *Machine Learnings* (ML) und der *Künstlichen Intelligenz* (KI). Dabei umfasst das ML Methoden

und Algorithmen zum Lernen von Mustern und Zusammenhängen aus Daten. Das RL fokussiert sich auf das Lernen von sinnvollen, situationsabhängigen Entscheidungen in vorliegenden Zuständen, um ein vom Anwender definiertes Ziel zu erreichen.

1.2 Zielsetzung und Eingrenzung

In dieser Dissertation soll die GHT mit Methoden der KI erweitert werden. Ziel ist es, die GHT durch RL so zu unterstützen, dass ein in der Praxis nutzbarer Mehrwert geschaffen wird. Dazu werden lateral belastete Crashstrukturen aus Aluminium, welche mittels Strangpressen hergestellt werden können, untersucht. Dabei stellen sich die folgenden Forschungsfragen, welche in dieser Dissertation diskutiert und beantwortet werden sollen:

- Welche zentralen Schnittstellen hat die GHT, bei denen KI-Methoden eingesetzt werden können?
- Welche dieser Schnittstellen können einen praktischen Mehrwert für die Entwicklung der GHT generieren?
- Wie lässt sich eine konkrete RL-Methodik zur Unterstützung der GHT formulieren?
- Inwiefern eignet sich RL gegenüber anderen KI-Methoden im Kontext der GHT?
- Wie gut kann die in dieser Arbeit umgesetzte RL-Methodik die GHT praktisch unterstützen?
- Welche Grenzen hat die umgesetzte RL-Methodik?

Obwohl es sich grundsätzlich aufgrund der Graphenrepräsentation der Entwürfe der GHT anbieten würde, werden in dieser Dissertation keine graphspezifischen ML-Algorithmen (z. B. Gori et al. 2005; Scarselli et al. 2009) untersucht, welche allgemeine Graphdaten direkt verarbeiten können. Ein wesentlicher Grund liegt darin, dass zum Zeitpunkt der Entwicklung der RL-Methodik dieser Dissertation noch keine Schnittstellen zu öffentlichen RL-Implementierungen mit dem geforderten Funktionsumfang vorhanden waren. Das wird im Verlauf der Dissertation weiter diskutiert.

Ebenso wird sich ausschließlich auf RL-Algorithmen fokussiert. Diese können ein ML-Modell trainieren, das komplexe Entscheidungen sequentiell treffen kann. Dabei werden in der Entscheidungsfindung auch erwartete, zukünftige Zustände berücksichtigt. Das hilft dem ML-Modell auch langfristig sinnvolle Entscheidungen zu tätigen. Alternative, ebenso populäre Algorithmen, wie beispielsweise der *NeuroEvolution of Augmented Topologies*-Algorithmus (NEAT-Algorithmus) (Stanley und Miikkulainen 2002), werden zwar angesprochen, aber nicht weiter untersucht. Der NEAT-Algorithmus kann je nach Aufgabenstellung in Konkurrenz zu RL-Algorithmen stehen. Dabei funktioniert dieser aber grundlegend anders. Aufgrund eines evolutionären Ansatzes, bei dem die

Architektur und Gewichte eines *Neuronalen Netzes* (NNs) evolviert werden, ist die Sample-Komplexität naturgemäß hoch. Dabei gibt die Sample-Komplexität an, wie viele Samples zum erfolgreichen Trainieren des ML-Modells nötig sind. Das ist aufgrund der benötigten ressourcenintensiven *Finite-Elemente*-Simulationen (FE-Simulationen) nicht praktikabel.

1.3 Aufbau der Dissertation

Diese Dissertation, deren Aufbau in Abbildung 1–1 dargestellt wird, sieht sich als Schnittstelle zwischen den Themengebieten der Crashmechanik aus dem Ingenieurwesen und des MLs mit Fokus auf das RL.

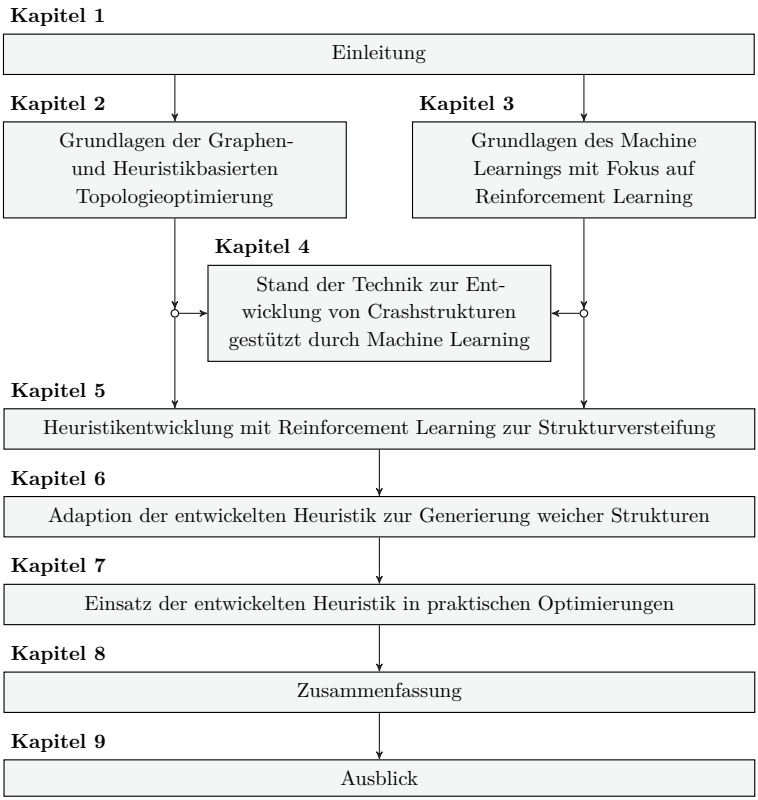


Abbildung 1–1: Aufbau der Dissertation

In Kapitel 2 werden die Grundlagen der Crashmechanik mitsamt der zugrundeliegenden numerischen Berechnungsmethodik und den aktuellen Ansätzen aus der Wissenschaft zur Topologieoptimierung crashbelasteter Strukturen vorgestellt. Ziel dieser ersten Abschnitte des Kapitels ist es, die für diese Dissertation relevanten Inhalte der GHT zusammenzustellen.

In Kapitel 3 folgt dann eine Einführung in die KI. Der Fokus liegt dabei auf dem ML und dem RL, da das RL integraler Bestandteil der in dieser Dissertation vorgestellten Methodik ist. Dabei werden alle essentiellen Grundlagen angesprochen und in dem für diese Arbeit nötigen Umfang erläutert.

Ein Überblick über den Stand der Technik aus der Themenkombination KI und der Auslegung von Crashstrukturen folgt in Kapitel 4. Dort wird eine möglichst breit aufgestellte Übersicht über die genannte Themenkombination gegeben. So dient das Kapitel als grundlegende Inspiration für die eigene Arbeit und erlaubt eine Abgrenzung zu der in dieser Dissertation entwickelten Methodik.

Bei der hier entwickelten Methodik handelt es sich um eine RL-basierte Heuristik, welche innerhalb der GHT zum Einsatz kommt und den Optimierungsprozess erweitern soll, um vielfältige und neue Entwurfsvorschläge innerhalb der Optimierung zu generieren und schlussendlich noch bessere Strukturen zu finden. In Kapitel 5 wird der Prozess von der Konzeptfindung der Heuristik über das Training der zugrundeliegenden ML-Modelle bis hin zur Integration der Heuristik in die GHT beschrieben. Fokus liegt hier auf der Entwicklung einer Heuristik, welche die Aufgabe hat, Strukturentwürfe lokal zu versteifen.

Konträr dazu werden in Kapitel 6 Konzepte für eine Heuristik auf Basis der in Kapitel 5 entwickelten Methodik vorgestellt, welche auch das Generieren weicher Strukturen ermöglichen sollen. Die Konzepte werden diskutiert und die Grenzen zur sinnvollen Nutzung aufgezeigt.

Die RL-basierte Heuristik zur lokalen Strukturversteifung wird in Kapitel 7 in praktischen Optimierungen auf ihren Nutzen hin untersucht. Dazu wird ein Rahmenmodell, ein Biegeträger und ein Schwellermodell hinsichtlich ihrer Steifigkeit mit der GHT in Crashlastfällen optimiert. Zusätzlich wird das Rahmenmodell bezüglich der Impaktorbeschleunigung mit der RL-basierten Heuristik optimiert. Die Optimierungsverläufe und -ergebnisse werden mit und ohne der neu entwickelten Heuristik miteinander verglichen.

Zuletzt werden in Kapitel 8 und Kapitel 9 eine Zusammenfassung und ein Ausblick gegeben. In der Zusammenfassung werden die zentralen Forschungsfragen aufgegriffen und beantwortet. Im Ausblick werden sinnvolle, möglicherweise an diese Dissertation anschließende Untersuchungen genannt, um die in dieser Dissertation erarbeitete Methodik weiter zu verbessern.

2 Grundlagen der Graphen- und Heuristikbasierten Topologieoptimierung

Die *Graphen- und Heuristikbasierte Topologieoptimierung* (GHT) ist eine Methode zur Form- und Topologieoptimierung crashbelasteter Strukturen. Zur Einordnung der Methode und der Begrifflichkeiten werden in den Abschnitten 2.1 bis 2.4 zunächst die für die GHT relevanten Themenfelder eingeführt. Zu den Themenfeldern gehören die Anforderungen an Crashstrukturen, die numerische Simulation von Crashstrukturen zur Evaluation der Strukturperformance, die Konzepte der automatisierten Strukturoptimierung zur systematischen Verbesserung der Struktur und ein Überblick über gängige Methoden zur Topologieoptimierung crashbelasteter Strukturen. In Abschnitt 2.5 wird die GHT detailliert beschrieben.

2.1 Anforderungen an Crashstrukturen

Eine effiziente und funktionsgerechte Auslegung crashbelasteter Fahrzeugstrukturen spielt eine große Rolle, um die Sicherheit von Fahrzeuginsassen und beteiligten Verkehrsteilnehmer*innen bestmöglich zu gewährleisten. Alle Maßnahmen zum Schutz der Beteiligten im Falle eines unvermeidbaren Unfalls gehören zur *passiven* Sicherheit von Fahrzeugen. Dazu zählen Systeme wie Rückhaltesysteme, Airbags, automatische Notrufsysteme und die für diese Dissertation primär relevanten, energieabsorbierenden Crashstrukturen. Gegensätzlich zu der passiven Sicherheit ist die Aufgabe der *aktiven* Fahrzeugsicherheit, die Entstehung eines Unfalls proaktiv zu vermeiden. Das wird u. a. mit Systemen wie dem *Antiblockiersystem* (ABS), dem *Elektronischen Stabilitätsprogramm* (ESP) und dem Notbremsassistenten erreicht.

Die Anforderungen an Crashstrukturen sind vielfältig und häufig gegensätzlich. Zum einen sollen die Crashstrukturen die strukturelle Integrität des Überlebensraums gewährleisten, um Quetschungen der Insassen zu vermeiden. Das wird i. d. R. mit steifen Strukturen erreicht. Zum anderen muss die kinetische Energie des Crashs durch die Strukturen aufgenommen werden. Das geschieht bei eher weichen Strukturen durch die Umwandlung der kinetischen Energie in innere Energie durch Deformationsarbeit. Dabei deformieren

sich die Strukturen plastisch. Die Art und Weise, wie die kinetische Energie durch die Struktur absorbiert wird, ist dabei für die Funktionsweise der Struktur entscheidend. Bereiche, die sich in einem Unfall gezielt plastisch deformieren sollen, sind Teil der sog. Knautschzone. Diese wurde bereits 1952 von Béla Barényi patentiert.

Zur Quantifizierung der geforderten Steifigkeit von verschiedenen Bereichen im Fahrzeug werden Lastniveaus bzw. Lastkorridore definiert. Diese sind ein häufiges Auslegungsziel für Crashkomponenten. Abbildung 2–1 zeigt qualitativ die Verteilung der Lastniveaus von Strukturen in einem Fahrzeug, die primär für einen Frontalcrash relevant sind.

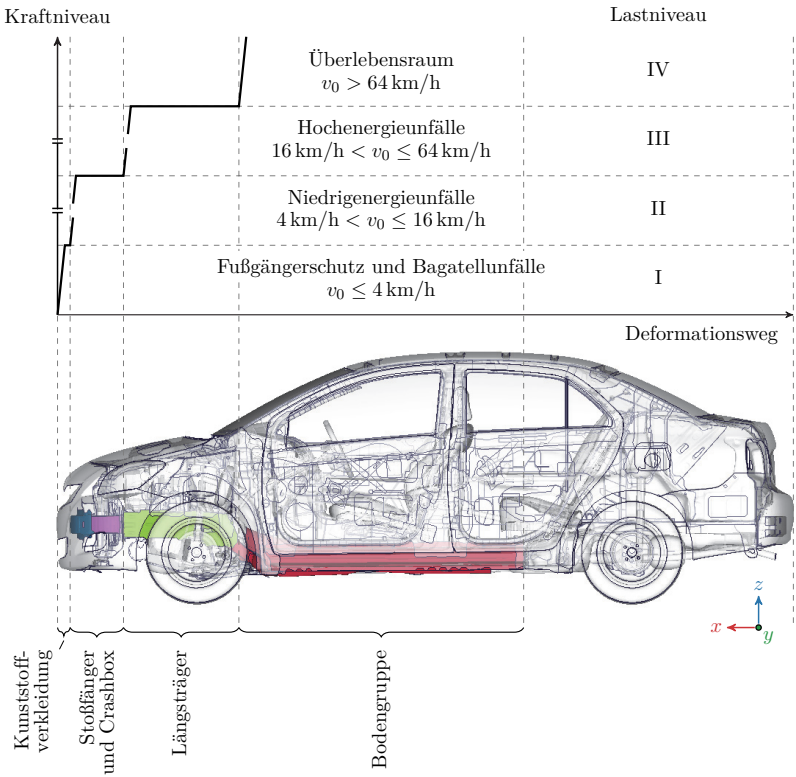


Abbildung 2–1: Qualitative Darstellung der crashrelevanten Lastniveaus und Strukturen im primären Lastpfad eines Toyota Yaris[1] in einem Frontcrash (modifiziert aus Schwanitz (2016))

[1]Modell verfügbar unter https://www.nhtsa.gov/crash-simulation-vehicle-models (Zuletzt aufgerufen am 15.09.2023)

Über verschiedene Komponenten hinweg wird eine progressive Steigerung des Lastniveaus und damit der Struktursteifigkeit von der Fahrzeugfront zum Überlebensraum des Fahrzeugs hin angestrebt. Das ermöglicht eine sequentielle Strukturdeformation von Komponenten in der Front des Fahrzeugs bis hin zum Überlebensraum. So bleiben im Falle eines Aufpralls mit niedriger Energie weiter hinten liegende Strukturen undeformiert und müssen nicht kostenintensiv ausgetauscht werden.

Die Performance einer Crashstruktur kann durch verschiedene Kriterien beschrieben werden. Neben den bereits genannten Lastniveaus und geforderten Struktursteifigkeiten, die mechanisch direkt als Auslegungskriterium herangezogen werden können, werden auch Verletzungskennwerte genutzt (z. B. CARHS GmbH 2023). Diese geben eine Abschätzung über die Verletzungsgefahr eines Insassen. Exemplarisch wird der *Head Injury Criterion*-Wert (HIC-Wert) vorgestellt, der die Schwere eine Kopfverletzung durch einen Unfall abschätzt. Der HIC-Wert ist ein beschleunigungsbasierter Kennwert, der den kritischen Beschleunigungsverlauf des Insassenkopfes einbezieht und sich nicht lediglich auf eine einzelne, maximale Kopfbeschleunigung stützt. Die Formel zur Berechnung des HIC-Wertes lautet

$$HIC = \max_{t_1,t_2} \left\{ (t_2 - t_1) \cdot \left[\frac{1}{t_2 - t_1} \int_{t_1}^{t_2} a(t) \mathrm{d}t \right]^{2.5} \right\}. \tag{2–1}$$

Dabei spannen t_1 und t_2 das Auswertezeitintervall auf und $a(t)$ ist der Kopfbeschleunigungsverlauf und wird als Vielfaches der Erdbeschleunigung g angegeben.

Neben den zentralen Anforderungen an Crashstrukturen, die direkten Einfluss auf die mechanische Verletzung der Unfallbetroffenen haben, müssen Crashstrukturen auch weitere Auslegungsziele erfüllen, um Folgeschäden zu vermeiden. Das sind beispielsweise Brandschutzmaßnahmen, wie das Sicherstellen der Dichtigkeit von kraftstoffführenden Leitungen und der Schutz der Batterie in Elektrofahrzeugen vor eindringenden Gegenständen.

2.2 Finite-Elemente-Methode im Kontext von Crashsimulationen

Die FEM ist eine numerische Methode, um komplexe physikalische Problemstellungen aus unterschiedlichsten Bereichen, wie beispielsweise der Strukturmechanik, der Strömungsmechanik, der Akustik und der Elektrotechnik zu simulieren. Das ermöglicht einen schnelleren und kosteneffizienteren Produktentwicklungsprozess gegenüber einem traditionellen, prototypenbasierten Prozess.

Für diese Dissertation ist ausschließlich die Lösung von strukturmechanischen Problemen relevant. Dazu wird in der FEM die Bauteilgeometrie diskretisiert, also in eine finite Anzahl an Elemente unterteilt, die miteinander über Knoten gekoppelt sind. Durch

die Reduktion des kontinuierlichen Problems auf die diskrete Formulierung lassen sich komplexe mechanische Probleme mathematisch leichter lösen. Wichtig ist anzumerken, dass die FEM auf der physikalischen und mathematischen Modellierung einer realen Problemstellung basiert. Entsprechend ist die Aussagekraft der Lösung stets ingenieurmäßig einzuordnen (Klein 2015).

Für lineare und statische Problemstellungen der Strukturmechanik lautet die zu lösende finite Grundgleichung

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}. \tag{2–2}$$

Dabei ist $\boldsymbol{K}$ eine lineare Steifigkeitsmatrix, $\boldsymbol{u}$ der gesuchte Verschiebungsvektor und $\boldsymbol{f}$ ein aufgeprägter, bekannter Lastvektor. Der Verschiebungsvektor lässt sich durch Multiplikation der inversen Steifigkeitsmatrix $\boldsymbol{K}^{-1}$ mit den bekannten Lasten bestimmen. Eine Problemstellung ist dann linear, wenn die zu ermittelnden Verschiebungen bzw. Deformationen klein sind, die Beziehung zwischen Spannung und Dehnung linear ist und kein Kontakt zwischen Komponenten auftritt. Bei einer statischen Problemstellung sind die aufgebrachten Lasten zeitlich konstant, sodass keine zeitabhängigen Effekte wie Trägheit oder Dämpfung auftreten. Viele ingenieurtechnische Probleme lassen sich basierend auf der o. g. Grundgleichung lösen.

Crashsimulationen sind konträr dazu dynamisch und hochgradig nichtlinear. Beschrieben werden sie durch die Bewegungsgleichung

$$\boldsymbol{M}\big(\boldsymbol{u}(t)\big)\ddot{\boldsymbol{u}}(t) + \boldsymbol{D}\big(\boldsymbol{u}(t)\big)\dot{\boldsymbol{u}}(t) + \boldsymbol{K}\big(\boldsymbol{u}(t)\big)\boldsymbol{u}(t) = \boldsymbol{f}\big(\boldsymbol{u}(t)\big). \tag{2–3}$$

Die lineare Grundgleichung (2–2) wird hier um Trägheits- und Dissipationseffekte zu einer nichtlinearen Differentialgleichung zweiter Ordnung erweitert. Die Trägheit des Systems wird durch die Massenmatrix $\boldsymbol{M}$ beschrieben und die Dissipationseffekte werden in Form einer geschwindigkeitsproportionalen Dämpfung durch die Dämpfungsmatrix $\boldsymbol{D}$ beschrieben. Analog zu Gleichung (2–2) ist $\boldsymbol{K}$ die Steifigkeitsmatrix. Die Matrizen $\boldsymbol{M}$, $\boldsymbol{D}$ und $\boldsymbol{K}$ werden Systemmatrizen genannt. $\ddot{\boldsymbol{u}}$ ist der Beschleunigungsvektor, $\dot{\boldsymbol{u}}$ ist der Geschwindigkeitsvektor und $\boldsymbol{u}$ der Verschiebungsvektor des Systems. Im Gegensatz zu der linearen Grundgleichung hängen $\boldsymbol{K}$ und $\boldsymbol{f}$, sowie zusätzlich $\boldsymbol{M}$ und $\boldsymbol{D}$, von dem betrachteten Verformungszustand $\boldsymbol{u}(t)$ zum Zeitpunkt t ab. Die Lösung der nichtlinearen Bewegungsgleichung lässt sich durch Integration dieser bestimmen. Für weiterführende mathematische Zusammenhänge zu Integrationsverfahren wird auf Literatur von Wriggers (2001), Klein (2015) und Wagner (2019) verwiesen.

Die für Crashstrukturen relevanten Nichtlinearitäten können in die folgenden drei Kategorien unterteilt werden (Wriggers 2001):

- *Geometrische Nichtlinearität* – Das System kann bei großen Verschiebungen und Rotationen nicht mehr ausschließlich auf die Ausgangskonfiguration bezogen werden, da es entsprechend Gleichung (2–3) von seinem aktuellen Verformungszustand

abhängig ist. Verzweigungsprobleme, auch Bifurkationen genannt, sind ebenfalls eine geometrische Nichtlinearität. Bei solchen Problemen ist die Eindeutigkeit einer Lösung nicht immer gegeben. Solche Phänomene treten häufig bei geometrischen Instabilitäten auf. Wriggers (2001) und Schumacher (2020) führen einen Knickstab als Beispiel für Bifurkationen an, da die Knickrichtung des Stabs in einem idealen System nicht eindeutig ist.

- *Materielle Nichtlinearität* – Typische Materialien von Crashstrukturen, wie Stahl und Aluminium, besitzen nur unter elastischer Deformation ein lineares Spannungs-Dehnungs-Verhalten. Im Crash soll durch große Deformationen und damit der Plastifizierung des Materials Energie absorbiert werden, sodass eine Modellierung des Materialverhaltens über den elastischen Bereich hinaus entscheidend ist. Zusätzlich ist das Materialverhalten abhängig von der anliegenden Dehnrate.
- *Nichtlinearität infolge von Randbedingungen* – Bei Kontaktphänomenen, wie sie für Crashsimulationen regelmäßig auftreten, wird das Eindringen zweier Strukturen ineinander verhindert. Durch den Kontakt wirkt eine Kraft, die das Eindringen beider Strukturen unterbindet. Damit sind die Randbedingungen abhängig vom Deformationszustand. Eine weitere Abhängigkeit vom Deformationszustand sind sog. Folgelasten. Diese greifen beispielsweise stets orthogonal zur Oberfläche der Struktur an und folgen ihr somit.

2.3 Grundlagen der automatisierten Strukturoptimierung

Methoden zur Optimierung mechanischer Strukturen erlauben es systematisch und automatisiert definierte Eigenschaften von Strukturen unter Einhaltung von Bedingungen zu verbessern, um deren Effizienz in ihren Einsatzgebieten zu steigern. Die zu optimierende(n) Eigenschaft(en) wird/werden im Kontext der Strukturoptimierung *Zielfunktion(en)* und die einzuhaltende(n) Bedingung(en) *Restriktion(en)* genannt. Für einen weiteren Überblick über die wesentlichen Begriffe der Strukturoptimierung werden diese in Tabelle 2–1 aufgelistet.

Eine Optimierung beginnt mit einem Startentwurf, welcher von dem/der Anwender*in vorgegeben wird. Dieser Entwurf wird auf Basis eines Berechnungs- bzw. Analysemodells berechnet. Eine Auswertung des berechneten Entwurfs erlaubt nun, die Performance der Struktur zu quantifizieren. Für den Fall, dass die Auswertung ergibt, dass es sich bei der Struktur um ein Optimum handelt, wurde der optimale Entwurf gefunden und die Optimierungsschleife wird verlassen. In praktischen Aufgabenstellungen ist meist nicht bekannt, ob das (theoretische) Optimum der Entwurfsvariablen gefunden wurde. Statt einer Überprüfung der Optimalität des Entwurfs lassen sich hier auch Abbruch- und/oder Konvergenzkriterien definieren. Dann spricht man von einer optimierten statt einer optimalen Struktur. Für den Fall, dass das Optimum nicht erreicht ist, werden

Tabelle 2–1: Wesentliche Begriffe der Strukturoptimierung (modifiziert aus Schumacher (2020))

Optimierungsalgorithmus	Mathematisches Verfahren zur Minimierung einer Zielfunktion mit/ohne Berücksichtigung von Restriktionen
Optimierungsverfahren	Zusammenstellung der Optimierungsansätze und Optimierungsalgorithmen zur Lösung von Optimierungsaufgaben
Optimierungsprozedur	Software zur Behandlung einer Optimierungsaufgabe, auch Optimierungsprogrammsystem genannt
Optimierungsstrategie	Vorgehensweise zur Reduktion komplexer Optimierungsaufgaben auf einfache Ersatzprobleme, die mit einem Optimierungsalgorithmus zu lösen sind
Zielfunktion(en)	Mathematische Formulierung eines oder mehrerer Konstruktions- bzw. Auslegungsziele
Restriktionen	Mathematisch formulierte Forderungen an die Konstruktion (einzuhaltende Bedingungen)
Analysemodelle	Mathematische Beschreibung der Modelleigenschaften (In der Strukturmechanik ist es das Strukturmodell, allgemein kann es auch als Simulationsmodell bezeichnet werden.)
Zustandsvariablen	Antworten des Analyse- bzw. Simulationsmodells
Entwurfsvariablen	Zu variierende Konstruktionsgrößen
Entwurfsraum	Bereich, in dem die Optimierung durchgeführt werden soll. Er wird in der Regel durch die Festlegung der unteren und oberen Grenzen der Entwurfsvariablen festgelegt.
Startentwurf	Startwerte der Entwurfsvariablen

die Entwurfsvariablen nach den Rechenvorschriften eines Optimierungsalgorithmus angepasst. Das Ziel ist, dass die nächste Auswertung der Berechnungsergebnisse auf dem Analysemodell näher am Optimum liegt. So ermöglicht der Optimierungsalgorithmus eine schrittweise Näherung an das zu findende Optimum. Je nach Aufgabenstellung eignen sich dafür direkte Suchmethoden, gradientenbasierte Algorithmen, evolutionäre Ansätze oder heuristische Verfahren. Schumacher (2020) stellt einige praxisrelevante Algorithmen zur Strukturoptimierung vor, auf die hier nicht weiter eingegangen wird. Der grundlegende Ablauf einer Strukturoptimierung lässt sich auf die in Abbildung 2–2 gezeigte Optimierungsschleife zurückführen.

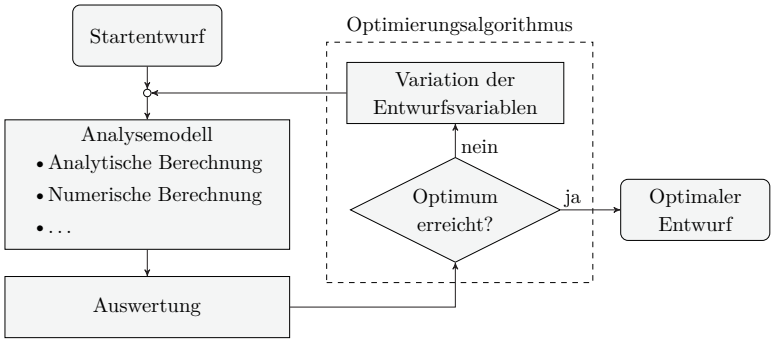


Abbildung 2–2: Optimierungsschleife (modifiziert aus Schumacher (2020))

Je nach Art der Entwurfsvariablen lassen sich Optimierungsaufgaben in verschiedene Klassen einteilen. Nach Schumacher (2020) unterscheiden sich die Strukturoptimierungsaufgaben wie folgt:

- Unter *Wahl der Bauweise* wird die Wahl der grundlegenden Gestaltung bzw. Konstruktion einer Struktur verstanden.
- Die *Wahl der Materialeigenschaften* beschreibt die Optimierung von Materialparametern.
- Mit der *Topologieoptimierung* wird die Lage und Anordnung von Strukturelementen optimiert. Dazu werden in der Optimierung wichtige Strukturelemente eingebracht und irrelevante Strukturelemente entfernt.
- Bei der *Formoptimierung* wird der Bauteilrand der vorhandenen Struktur geometrisch angepasst. Im Gegensatz zur Topologieoptimierung werden hierbei keine Strukturelemente hinzugefügt oder entfernt.
- Die *Dimensionierung* beschreibt Entwurfsvariablen, wie Wanddicken und Querschnittsdimensionen.

Die verschiedenen Klassen sind in Abbildung 2–3 visualisiert. Je nach Optimierungsverfahren ist es auch möglich, dass mehrere der genannten Klassen zusammen optimiert werden. Ein Beispiel hierfür ist die Kombination aus Topologieoptimierung und Formoptimierung. Hier wird dann Material an irrelevanten Stellen aus dem Entwurfsraum entfernt, sodass Hohlräume entstehen. Diese Hohlräume selbst werden dann in ihrer Form zusätzlich verändert.

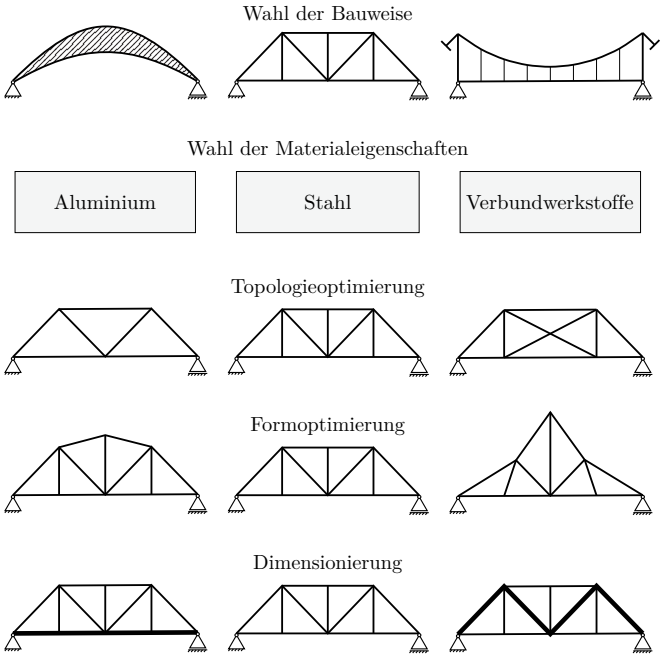


Abbildung 2–3: Klassifizierung von Strukturoptimierungsaufgaben nach Art der Entwurfsvariablen (modifiziert aus Schumacher (2020))

Ein Optimierungsproblem lässt sich wie folgt formulieren (Schumacher 2020):

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}), \\ \text{so dass} \quad & g_j(\boldsymbol{x}) \leq 0, \\ & h_k(\boldsymbol{x}) = 0, \\ & x_i^{(l)} \leq x_i \leq x_i^{(u)}. \end{aligned}$$

Dabei ist $\boldsymbol{x}$ der Vektor der Entwurfsvariablen und $f(\boldsymbol{x})$ die von den Entwurfsvariablen abhängige Zielfunktion. $g_j(\boldsymbol{x})$ mit $j \in \{1, \ldots, n_g\}$ sind n_g unterschiedliche Ungleichheitsrestriktionen, deren Einhaltung für alle Werte $g_j \leq 0$ gewährleistet ist. $h_k(\boldsymbol{x}) = 0$ mit $k \in \{1, \ldots, n_h\}$ sind n_h unterschiedliche Gleichheitsrestriktionen, welche nur im Fall $h_k = 0$ eingehalten werden. Bei beiden Restriktionen handelt es sich um implizite Restriktionen. Die explizite Restriktion $x_i^{(l)} \leq x_i \leq x_i^{(u)}$ schränkt den Definitionsbereich von der i-ten Entwurfsvariable auf das Intervall $\left[x_i^{(l)}; x_i^{(u)} \right]$ ein. Dadurch wird der Entwurfsraum für das Optimierungsproblem festgelegt.

2.4 Methoden zur Topologieoptimierung crashbelasteter Strukturen

Die Topologieoptimierung crashbelasteter Strukturen erweist sich als Herausforderung für Forschung und Industrie. Die Ursachen dafür sind in der in Abschnitt 2.2 diskutierten Komplexität von Crashsimulationen und der Crashmechanik zu finden, wodurch es schwierig und häufig auch ineffizient ist, nutzbare Sensitivitäten für mathematische, gradientenbasierte Optimierungsalgorithmen zu bestimmen.

Eine frühe Arbeit der Topologieoptimierung von Crashstrukturen von Mayer et al. (1996) kommt ohne direkten Einsatz von Sensitivitäten, also den Ableitungen der Entwurfsvariablen nach der Zielfunktion und den Restriktionen, aus. Die Beschreibung der Entwurfsvariablen und der Topologie folgt dabei der *Voxel*-Methode (Bendsøe und Kikuchi 1988; Bendsøe und Sigmund 2004), welche bereits bei der Topologieoptimierung von linear statischen Strukturen eingesetzt wird. Dazu wird der Entwurfsraum in Voxel, das sind kleine Bereiche im Entwurfsraum, aufgeteilt. In der erweiterten Variante von Rozvany et al. (1992) werden jedem Voxel Materialdichten zwischen 0 (kein Material) und 1 (Vollmaterial) zugewiesen. Basierend auf dieser Beschreibung zeigt Mayer et al. (1996), dass die Energieabsorption der Struktur durch Anpassung der Voxeldichten auf Basis eines Optimalitätskriteriums zu gewichteten Auswertezeitpunkten maximiert werden kann. (Schumacher 2020)

Die *Hybrid Cellular Automaton*-Methode (HCA-Methode) (Patel 2007; Patel et al. 2009) basiert auf dem aus der Informatik bekannten Prinzip der zellulären Automaten (Wolfram 2002). Dabei modellieren zelluläre Automaten dynamische Systeme, bei denen die Zellzustände aus der direkten Nachbarschaftsbeziehung einer betrachteten Zelle ermittelt werden. Die HCA-Methode ist ebenfalls voxelbasiert und verwendet ein Optimalitätskriterium. Der zelluläre Automat verteilt das Material der Struktur dabei heuristisch so um, dass jeder Voxel für den gegebenen Lastfall gleichermaßen ausgenutzt wird. Dafür wird die innere Energiedichte der Voxel als Kriterium herangezogen. Voxel mit Zwischendichten wird eine skalierte Spannungs-Dehnungs-Kurve zugewiesen, um das Materialverhalten näherungsweise abbilden zu können. Um auch crash-typische, dünnwandige Blechstrukturen effizient mit der HCA-Methode optimieren zu können, kann die Erweiterung *Hybrid Cellular Automaton for Thin-Walled Structures* (HCA-TWS) (Hunkeler 2014; Zeng 2019) herangezogen werden. Dort wird die Entwurfsdomäne anstelle von Voxeln durch dünnwandige Strukturen gefüllt, welche die Topologie der Struktur beschreiben.

Nachteile der voxelbasierten Ansätze von Mayer et al. (1996) und Patel (2007) sind nach Schumacher (2020)

- die durch die Voxelform schwierige Abbildung von Kontaktphänomenen,
- die hohe Rechenzeit durch benötigte Volumenelemente, wobei Crashstrukturen typischerweise schalenartig sind und
- die Einschränkung der Ziel- und Restriktionsfunktionen auf Verschiebungs-, Massen- und Kraftfunktionen aufgrund von fehlenden Sensitivitätsinformationen.

Weiterhin führt Sperber (2022) an, dass der Ausgleich der Energiedichte für Crashprobleme fragwürdig ist, da viele effiziente Crashstrukturen keine homogene Verteilung der inneren Energie aufweisen.

Die *Evolutionary Level-Set-Method* (EA-LSM) (Bujny et al. 2018; Bujny 2020) ist ein aktueller Forschungsansatz zur Topologieoptimierung crashbelasteter Strukturen. Level-Set-Methoden definieren die Struktur über eine parametrisierte Level-Set-Funktion $\Phi(\boldsymbol{x})$, wobei $\boldsymbol{x}$ die Position im Bauraum beschreibt. Für den Fall, dass $\Phi(\boldsymbol{x}) > 0$ ist, ist Material an der Position $\boldsymbol{x}$ vorhanden. Im Umkehrschluss ist für den Fall $\Phi(\boldsymbol{x}) < 0$ kein Material an $\boldsymbol{x}$ vorhanden. Der Spezialfall $\Phi(\boldsymbol{x}) = 0$ definiert den Rand der Struktur, also den Übergang zwischen vorhandenem und keinem Material. Die Parameter der Level-Set-Funktion werden dann direkt von einem evolutionären Optimierungsalgorithmus angepasst. Um die Anzahl an benötigten Funktionsaufrufen der Methode zu reduzieren, wird diese durch Methoden des MLs erweitert. Dazu soll der Optimierer beispielsweise durch approximierte Gradienteninformationen gelenkt werden oder die Zielfunktionen und Restriktionen der Optimierungsaufgabe durch Ersatzfunktion abgebildet werden, um FE-Simulationen einzusparen.

Bei der *Equivalent Static Loads Method* (ESLM) (Choi und Park 2002; Park 2011) handelt es sich um einen Ansatz, bei dem sog. äquivalente Lasten aus nichtlinearen dynamischen FE-Simulationen in der Art berechnet werden, dass in einer linearen FE-Simulation des Modells gleiche Verformungen erwirkt werden. Das wird für mehrere Zeitpunkte des Crashs durchgeführt, so dass auf dem linearen FE-Modell mehrere Lastfälle berechnet werden. Für dieses lineare Ersatzproblem kann dann eine gradientenbasierte Topologieoptimierung durchgeführt werden. Ein wichtiger Schritt ist die anschließende Überführung des Voxelmodells in ein crash-typisches Schalenmodell, um in der nächsten Iteration neue Ersatzlasten aus der Simulation des neuen Crashmodells ableiten zu können. Das Vorgehen bei der ESLM ist in Abbildung 2–4 zusammengefasst.

Triller et al. (2021) stellen eine Erweiterung der ESLM namens *Difference-based Equivalent Static Loads* (DiESL) vor, bei der nicht die undeformierte Struktur als Basis für die Topologieoptimierung des linearen FE-Modells dient, sondern die deformierte Struktur zu verschiedenen Zeitpunkten der nichtlinearen Analyse. So müssen die äquivalenten, statischen Lasten für die Topologieoptimierung des linearen FE-Modells lediglich den

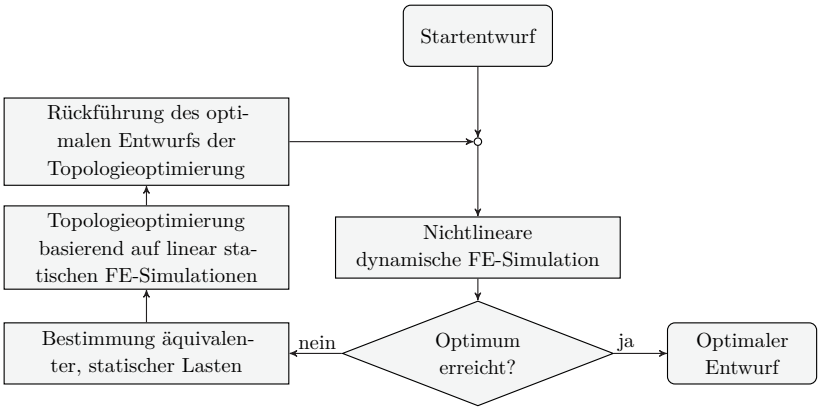


Abbildung 2–4: Optimierungsschleife der ESLM (modifiziert aus Schumacher (2020))

Deformationszustand des Folgezeitpunkts erwirken. Das erlaubt nach Aussage der Autoren eine signifikant verbesserte Approximationsgüte und schnellere Konvergenz im Vergleich zur konventionellen ESLM.

Clemens und Schumacher (2023) stellen eine ESLM-Alternative zur Optimierung tiefziehbarer, crashbelasteter Schalenstrukturen vor. In Crashsimulationen werden Kontaktkräfte ermittelt, die, stellvertretend zu den äquivalenten Lasten der ESLM, dem zugehörigen linear statischen Modell in einer Topologieoptimierung aufgeprägt werden. Diese Topologieoptimierungen basieren auf Voxelmodellen, die für die Crashsimulationen in Schalenmodelle übersetzt werden.

Bei dem *Ground Structure Approach* (GSA) (Pedersen 2003; Pedersen 2004) handelt es sich um einen Ansatz, bei dem der Bauraum der Struktur durch eindimensionale Balkenelemente in einer vom Nutzer vorgegebenen Anordnung, der Grundstruktur, ausgefüllt ist. Die Balkenelemente berücksichtigen geometrische Nichtlinearitäten und besitzen plastische Gelenke. Damit sind die Balkenelemente für große Deformationen geeignet. Der Optimierungsalgorithmus ist gradientenbasiert und erhält analytische Gradienten auf Basis von impliziten und quasi-statischen FE-Simulationen. Einer der größten Nachteile der Methode ist nach Bujny (2020) die Abhängigkeit des Optimierungsergebnisses von der Grundstruktur.

Methoden zur Bestimmung nutzbarer Sensitivitäten existieren für moderat nichtlineare Problemstellungen. Weider (2021) stellt eine Methode zur Bestimmung topologischer Sensitivitäten für Optimierungsaufgaben mit materiellen und geometrischen Nichtlinearitäten vor. Eine topologische Sensitivität gibt dabei an, wie groß der Einfluss eines infinitesimal kleinen Lochs oder Hohlraums in der Struktur auf die Zielfunktion ist. Berechnet werden die topologischen Sensitivitäten mit der adjungierten Methode und einer

Interpolation des elasto-plastischen Materialverhaltens. Ivarsson et al. (2018) berechnen ebenfalls nichtlineare Sensitivitäten mit der adjungierten Methode auf Basis eines linear isotropischen Materialmodells mit Verfestigung.

Zuletzt ist die *Graphen- und Heuristikbasierte Topologieoptimierung* (GHT) zu nennen, welche durch konkurrierende heuristische Topologieänderungen die Struktur verbessert und dadurch nicht direkt abhängig von Sensitivitätsinformationen ist. Aufgrund der Relevanz der GHT für diese Dissertation, wird diese im folgenden Abschnitt 2.5 detailliert vorgestellt.

2.5 Graphen- und Heuristikbasierte Topologieoptimierung

Die *Graphen- und Heuristikbasierte Topologieoptimierung* (GHT) (Olschinka und Schumacher 2008; Ortmann und Schumacher 2013; Ortmann 2015; Ortmann et al. 2021) ist eine Methode zur Topologie- und Formoptimierung sowie Dimensionierung von crashbelasteten Strukturen. Erstes Anwendungsfeld der Methode war die Optimierung zweidimensionaler Querschnitte von strang-gepressten Profilstrukturen in lateralen Crashlastfällen. Durch kontinuierliche Forschung wurde die GHT auch auf Blechstrukturen (Link et al. 2018), gewickelte Kompositstrukturen (Schneider et al. 2019; Schneider 2023) und auf die Layoutfindung für dreidimensionale Strukturen (Beyer et al. 2021) erweitert. Ebenso verfügt die GHT nun auch über Verfahren zur Optimierung von Strukturen in axialen Lastfällen (Sperber 2022), wie es beispielsweise bei der Auslegung von Crashboxen der Fall ist.

2.5.1 Geometriebeschreibung durch Graphen

Ein zentraler Bestandteil der GHT ist die Beschreibung des Profilquerschnitts durch einen mathematischen Graphen, welcher Strukturgraph G_S genannt wird. Dieser ermöglicht eine einfache Beschreibung und Anpassung der Form und Topologie der Struktur durch Hinzufügen und Löschen von Wänden, insbesondere gegenüber einer direkten Anpassung des FE-Netzes. Aus der Graphenbeschreibung wird dann automatisch eine FE-Struktur der zu optimierenden Komponente abgeleitet, welche in ein bereits definiertes und zugrundeliegendes FE-Modell mitsamt Lastfall automatisiert eingebaut wird.

Graphen G bestehen aus einer Menge Vertices $\mathcal{V}^{\{G\}}$ und einer Menge Kanten $\mathcal{E}^{\{G\}}$, die diese Vertices miteinander verbinden. Die Vertices in einem Strukturgraphen spannen den Querschnitt des Extrusionsprofils auf und die Kanten definieren, zwischen welchen Vertices Wände des Extrusionsprofils entstehen sollen. Bei Strukturgraphen sind in dieser Dissertation sowohl die Vertices $v_i \in \mathcal{V}^{\{G_\mathrm{S}\}}$ als auch die Kanten $e_i \in \mathcal{E}^{\{G_\mathrm{S}\}}$ mit *Identifikationsnummern* (IDs) versehen. So können die zugehörigen Vertices und Kanten direkt von der GHT angesprochen werden. In Extrusionsrichtung wird der Querschnitt

von der GHT als konstant angenommen. Die Strukturgraphen sind planar, d. h., dass sich keine Kanten direkt schneiden. Wenn sich Kanten schneiden sollen, werden die Schnitte durch das Einbringen neuer Vertices an der Schnittstelle aufgelöst. Ein Beispielgraph ist in Abbildung 2–5 dargestellt.

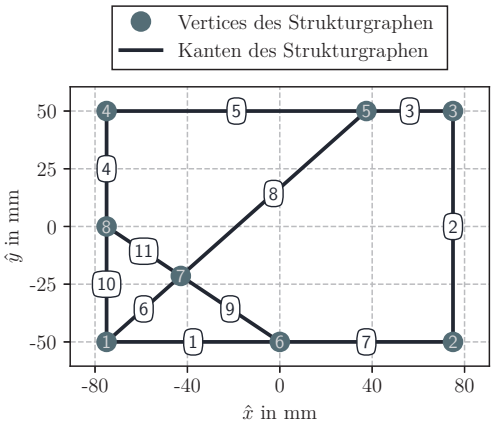


Abbildung 2–5: Beispiel eines Strukturgraphen G_S der GHT mit zugehörigen IDs der Vertices $\mathcal{V}^{\{G_\mathrm{S}\}}$ und Kanten $\mathcal{E}^{\{G_\mathrm{S}\}}$

Für eine möglichst einfache Manipulation der Strukturgraphen beschreibt die GHT diese über ein *American Standard Code for Information Interchange*-basiertes (ASCII-basiertes) Austauschformat. Die zu Abbildung 2–5 zugehörige Beschreibung in der GHT-Syntax mit den wichtigsten Schlüsselwörtern wird in Abbildung 2–6 dargestellt.

In Zeile 1 wird spezifiziert, dass im Folgenden ein Graph definiert wird, welcher einen zweidimensionalen Querschnitt für ein Extrusionsprofil repräsentiert. Anschließend werden in den Zeilen 2 bis 9 die acht Vertices des Strukturgraphen definiert. Neben einer ID erhalten die Vertices auch Raumkoordinaten. Die Koordinaten sind lokale Profilkoordinaten in $\hat{x}$-, $\hat{y}$- und $\hat{z}$-Richtung. Die $\hat{z}$-Komponente entspricht der Extrusionsrichtung, weshalb alle Vertices des Querschnitts innerhalb einer Ebene bei $\hat{z} = 0$ definiert sind. Alle einheitenbehafteten Werte in der Graphenbeschreibung besitzen die jeweilige Einheit, in der auch das zugrundeliegende FE-Modell aufgebaut ist. In den Zeilen 10 bis 20 werden die elf Kanten des Strukturgraphen definiert. Auch diese erhalten eine ID. Zusätzlich werden die zwei Vertices angegeben, die die Kante aufspannen. Jede Kante erhält eine individuelle Blechdicke und referenziert die Material-ID des FE-Modells. Der Parameter FIX steuert, ob eine Kante im Optimierungsprozess entfernt werden darf. Die Aufgabe des Parameters SPECIFIER ist es, eine Verknüpfung zwischen den Graphkanten und den zugehörigen FE-Strukturkomponenten herzustellen.

```
GRAPH;  1; TYPE(2DEXTRUSION);
VERTEX; 1; COORDINATES(-75.00, -50.00, 0.00)
VERTEX; 2; COORDINATES( 75.00, -50.00, 0.00)
VERTEX; 3; COORDINATES( 75.00,  50.00, 0.00)
VERTEX; 4; COORDINATES(-75.00,  50.00, 0.00)
VERTEX; 5; COORDINATES( 37.50,  50.00, 0.00)
VERTEX; 6; COORDINATES(  0.00, -50.00, 0.00)
VERTEX; 7; COORDINATES(-42.86, -21.43, 0.00)
VERTEX; 8; COORDINATES(-75.00,   0.00, 0.00)
EDGE;   1; VERTICES(1, 6); THICKNESS(1.4); MATERIAL(1);  FIX(true); SPECIFIER(K1)
EDGE;   2; VERTICES(2, 3); THICKNESS(1.4); MATERIAL(1); FIX(false); SPECIFIER(K2)
EDGE;   3; VERTICES(3, 5); THICKNESS(1.4); MATERIAL(1); FIX(false); SPECIFIER(K3)
EDGE;   4; VERTICES(4, 8); THICKNESS(1.4); MATERIAL(1);  FIX(true); SPECIFIER(K4)
EDGE;   5; VERTICES(5, 4); THICKNESS(1.4); MATERIAL(1); FIX(false); SPECIFIER(K5)
EDGE;   6; VERTICES(1, 7); THICKNESS(1.4); MATERIAL(1); FIX(false); SPECIFIER(K6)
EDGE;   7; VERTICES(6, 2); THICKNESS(1.4); MATERIAL(1);  FIX(true); SPECIFIER(K7)
EDGE;   8; VERTICES(7, 5); THICKNESS(1.4); MATERIAL(1); FIX(false); SPECIFIER(K8)
EDGE;   9; VERTICES(6, 7); THICKNESS(1.4); MATERIAL(1); FIX(false); SPECIFIER(K9)
EDGE;  10; VERTICES(8, 1); THICKNESS(1.4); MATERIAL(1);  FIX(true); SPECIFIER(K10)
EDGE;  11; VERTICES(7, 8); THICKNESS(1.4); MATERIAL(1); FIX(false); SPECIFIER(K11)
```

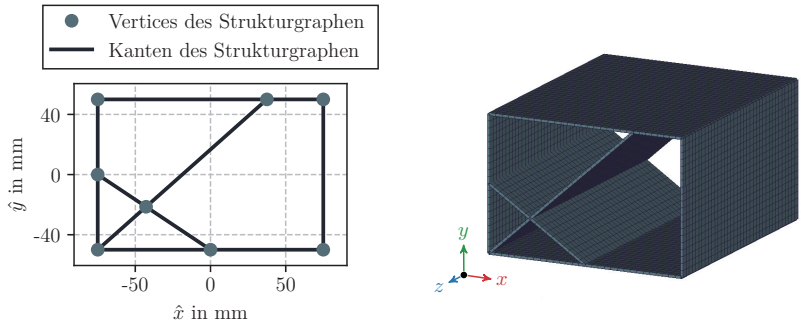
Abbildung 2–6: Syntax zur Beschreibung des Strukturgraphen aus Abbildung 2–5

2.5.2 Aufbau des Simulationsmodells

Die von der GHT zu optimierende Struktur wird aus dem Strukturgraphen abgeleitet und in ein bestehendes Simulationsmodell eingebaut. Das bestehende, unvollständige Simulationsmodell wird von dem/der Anwender*in zuvor definiert. Es beinhaltet Lasten und Randbedingungen, Simulationsparameter und ggfs. andere Strukturkomponenten, welche mit der zu optimierenden Struktur interagieren.

Grundlage für den Aufbau der zu optimierenden Struktur aus dem Strukturgraphen ist die Software *Graph and Mechanics Builder* (GRAMB), welche zum einen ein GHT-internes Vernetzungstool bereitstellt, aber zum anderen auch proprietäre Vernetzungstools ansteuern kann, um eine noch flexiblere und robustere Netzgenerierung zu ermöglichen. Diese Software analysiert den ASCII-basierten Strukturgraphen (vgl. Abbildung 2–6). Daraus leitet GRAMB ein FE-Netz aus Schalenelementen ab. In einer Konfigurationsdatei der GHT sind die zugehörigen Elementgrößen, Informationen zur Anzahl an Integrationspunkten je Schalenelement und einer vom Nutzer vorgegebenen Extrusionstiefe hinterlegt.

Dieses Netz kann dann in dem bestehenden, unvollständigen Simulationsmodell, welches für verschiedene Topologien und abgeleitete FE-Netze innerhalb einer Optimierung stets identisch ist, inkludiert werden. Damit ist das Simulationsmodell vollständig aufgebaut und rechenfähig. Abbildung 2–7 zeigt die Ableitung eines FE-Modells aus einem Strukturgraphen bei einer Extrusionstiefe von 180 mm und einer Elementkantenlänge von 5 mm.



(a) Strukturgraph in Profilkoordinaten **(b)** Zugehöriges FE-Modell

Abbildung 2–7: Beispiel eines Strukturgraphen, welcher mittels GRAMB in das zugehörige FE-Modell übersetzt wird

2.5.3 Fertigungsrestriktionen

Die Fertigungsrestriktionen der GHT sollen während der Optimierung sicherstellen, dass alle vorgeschlagenen Entwürfe auch durch Strangpressen praktisch herstellbar sind. Die Einhaltung der Fertigungsrestriktionen wird auf Basis des Strukturgraphen G_S ermittelt. In Tabelle 2–2 werden die einzuhaltenden Fertigungsrestriktionen aufgezeigt und in Abbildung 2–8 exemplarisch visualisiert.

Tabelle 2–2: Fertigungsrestriktionen in der GHT

Parameter	Restriktionsgrenze	Geltungsbereich
Kantenlänge $l_{e_i}^{\{G_\mathrm{S}\}}$	$l_{e_i}^{\{G_\mathrm{S}\}} \geq l_\mathrm{min}$	$\forall e_i \in \mathcal{E}^{\{G_\mathrm{S}\}}$
Wandstärke $t_{e_i}^{\{G_\mathrm{S}\}}$	$t_\mathrm{min} \leq t_{e_i}^{\{G_\mathrm{S}\}} \leq t_\mathrm{max}$	$\forall e_i \in \mathcal{E}^{\{G_\mathrm{S}\}}$
Abstand $d_{e_i,e_j}^{\{G_\mathrm{S}\}}$	$d_{e_i,e_j}^{\{G_\mathrm{S}\}} \geq d_\mathrm{min}$	$\forall e_i, e_j \in \mathcal{E}^{\{G_\mathrm{S}\}}$ mit $e_i \neq e_j \wedge \mathcal{V}^{\{e_i\}} \cap \mathcal{V}^{\{e_j\}} = \emptyset$
Winkel $\alpha_{e_i,e_j}^{\{G_\mathrm{S}\}}$	$\alpha_{e_i,e_j}^{\{G_\mathrm{S}\}} \geq \alpha_\mathrm{min}$	$\forall e_i, e_j \in \mathcal{E}^{\{G_\mathrm{S}\}}$ mit $e_i \neq e_j \wedge \mathcal{V}^{\{e_i\}} \cap \mathcal{V}^{\{e_j\}} \neq \emptyset$

Der Geltungsbereich zu den Fertigungsrestriktionen gibt an, für welche Kanten bzw. Kantenpaare die jeweilige Restriktion überprüft wird. Dabei ist $\mathcal{E}^{\{G_\mathrm{S}\}}$ die Menge aller Kanten des Strukturgraphen und $\mathcal{V}^{\{e_i\}}$ ist die Menge der Vertices, die die Kante e_i aufspannen. Die Kantenlänge $l_{e_i}^{\{G_\mathrm{S}\}}$ und Wandstärke $t_{e_i}^{\{G_\mathrm{S}\}}$ wird für jede Wand überprüft. Der Abstand $d_{e_i,e_j}^{\{G_\mathrm{S}\}}$ zwischen den Kanten e_i und e_j wird überprüft, solange es sich nicht um gleiche oder direkt verbundene Kanten handelt. Bei dem Winkel $\alpha_{e_i,e_j}^{\{G_\mathrm{S}\}}$ dürfen die Kanten e_i und e_j ebenfalls nicht identisch sein und müssen sich einen Knoten teilen und somit aneinandergrenzen.

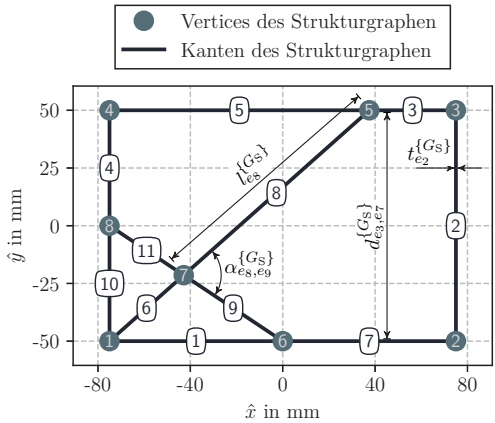


Abbildung 2–8: Darstellung der zu überprüfenden Fertigungsrestriktionen an exemplarischen Kanten

2.5.4 Vorstellung der Heuristiken

Die Heuristiken in der GHT steuern die Art und Weise, wie die Form und Topologie von Strukturgraphen auf Basis der zugehörigen FE-Simulationen angepasst werden. Heuristiken erlauben i. Allg. Aussagen über ein System mit begrenztem Wissen zu treffen. Somit lassen sich Heuristiken auch als kondensiertes Expertenwissen auffassen, welche im Kontext der GHT über Strukturanpassungen entscheiden.

Die entwickelten Heuristikregeln leiten sich u. a. aus einer Sammlung von über 150 Ideen aus einem Brainstorming mit Automobilherstellern ab (Schumacher und Ortmann 2013). Verschiedene Heuristiken haben dabei unterschiedliche Ziele, die die Strukturänderungen erreichen sollen. So gibt es Heuristiken, die die Struktur steifer bzw. weniger steif machen sollen und Heuristiken, die die Struktur vereinfachen sollen. Es gibt konkurrierende und nicht-konkurrierende Heuristiken. Konkurrierende Heuristiken treten gegeneinander an und werden bezüglich der Performance miteinander verglichen. Nicht-konkurrierende Heuristiken werden auf alle von den konkurrierenden Heuristiken vorgeschlagenen Entwürfen angewandt.

Im Folgenden werden die in dieser Arbeit genutzten, konkurrierenden Heuristiken vorgestellt. Die Erklärungen der Heuristiken basieren auf den Veröffentlichungen von Ortmann (2015), Ortmann et al. (2021) und Beyer et al. (2021). Jeder der im Folgenden vorgestellten Heuristik liegt eine mathematische Formel zugrunde, nach der die Heuristik Kenngrößen für die Kanten des Strukturgraphen bzw. Wände des zugehörigen FE-Modells

bestimmt und somit Aussage darüber treffen kann, wo das Einbringen einer neuen Kante in den Strukturgraphen am sinnvollsten ist. Details zu den zugrundeliegenden Formeln der Heuristiken und deren Evaluation sind u. a. in Schneider (2023) zu finden.

Die Heuristik *Delete Needless Walls* (DNW) (deutsch: *Löschen irrelevanter Wände*) kann Wände, die für die Erfüllung des geforderten strukturmechanischen Verhaltens nicht benötigt werden, aus der Struktur entfernen. Als Kriterium wird die innere Energiedichte einer betrachteten Wand im Verhältnis zur mittleren Energiedichte der Struktur herangezogen. Die Wand mit dem niedrigsten Verhältnis wird, sofern dieses unterhalb eines Grenzwertes liegt, von der Heuristik gelöscht. Wenn bei dem Löschprozess freie Wände mit nur einseitiger Verbindung zur Struktur entstehen, werden die freien Wände ebenfalls gelöscht. In Abbildung 2–9 wird das Funktionsprinzip der Heuristik anhand eines mit einem zylindrischen Impaktor beaufschlagten Extrusionsprofils dargestellt.

Das Energiekriterium ermöglicht in vielen Fällen eine plausible Entscheidung über die Wichtigkeit einer Wand in einem Lastfall. Es ist allerdings auch möglich, dass eine Wand nur wenig Energie aufnimmt und trotzdem für die Integrität der Struktur entscheidend ist. So könnte diese Wand ohne starke Deformation eine andere Wand gegen Knicken oder Beulen abstützen.

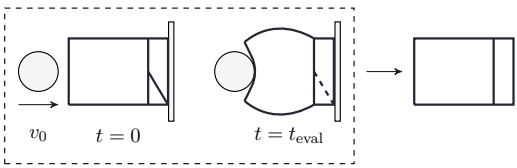


Abbildung 2–9: Funktionsprinzip der Heuristik DNW (modifiziert aus Ortmann et al. (2021))

Die Heuristik *Support Buckling Walls* (SBW) (deutsch: *Abstützen schnell deformierender Wände*) soll Wände detektieren und abstützen, die eine starke Knick- oder Beulneigung aufweisen. Dadurch soll ein gleichmäßiges Beulverhalten der Gesamtstruktur erzielt werden, um eine effiziente Energieabsorption der Struktur zu begünstigen. Das Funktionsprinzip der Heuristik wird in Abbildung 2–10 anhand einer Struktur zwischen zwei sich bewegenden, starren Wänden dargestellt. Als Kriterium wird ein normierter Knick- bzw. Beulindex bestimmt, welcher darauf beruht, dass sich FE-Knoten beim Knicken und Beulen schnell zueinander bewegen. Die Heuristik versucht die Wand mit dem größten Index mittig und orthogonal abzustützen.

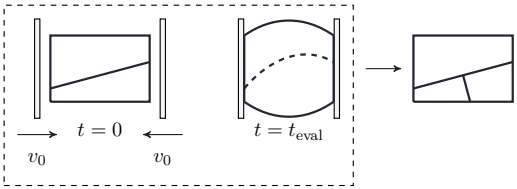


Abbildung 2–10: Funktionsprinzip der Heuristik SBW (modifiziert aus Ortmann et al. (2021))

Das Ziel der Heuristik *Balance Energy Density* (BED) (deutsch: *Ausgleichen der Energiedichte*) ist es, Bereiche der Struktur mit niedriger Energiedichte und Bereiche mit hoher Energiedichte zu verbinden, um eine homogene Verteilung der Energiedichte der Struktur zu erzielen. In Abbildung 2–11 wird das Funktionsprinzip der Heuristik aufgezeigt. Die Wand, auf die der zylindrische Impaktor trifft, absorbiert den größten Anteil der kinetischen Energie des Impaktors. Die gegenüberliegende Wand nimmt kaum Energie auf, da diese sich an der starren Wand abstützt. Die Heuristik verbindet die beiden Wände mittig miteinander. Zur Berechnung wird die innere Energiedichte jeder Wand in jedem Lastfall berechnet. Diese werden hier zusätzlich mit der maximalen Energiedichte einer Wand über möglicherweise mehrere Lastfälle gewichtet. Dadurch gehen Lastfälle mit höherer Energiedichte stärker in die Betrachtung ein.

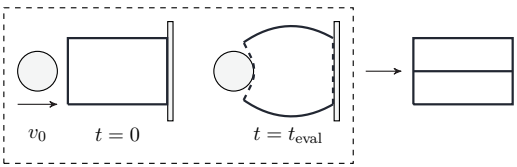


Abbildung 2–11: Funktionsprinzip der Heuristik BED (modifiziert aus Ortmann et al. (2021))

Die Heuristiken *Use Deformation Space Tension* (UDST) (deutsch: *Ausnutzen des Deformationsraums unter Zugbelastung*) und *Use Deformation Space Compression* (UDSC) (deutsch: *Ausnutzen des Deformationsraums unter Druckbelastung*) haben die Aufgabe den verfügbaren Deformationsraum möglichst effizient auszunutzen. Dazu wird zwischen Wänden, die sich relativ zueinander bewegen, eine weitere Wand eingebracht, um der Relativbewegung entgegen zu wirken und Energie zu absorbieren. Die Heuristik UDST bringt dabei eine neue Wand zwischen sich voneinander entfernenden Wänden ein und die Heuristik UDST bringt eine Wand zwischen sich aufeinander zubewegenden Wänden ein. Das Funktionsprinzip beider Heuristiken ist in Abbildung 2–12 dargestellt.

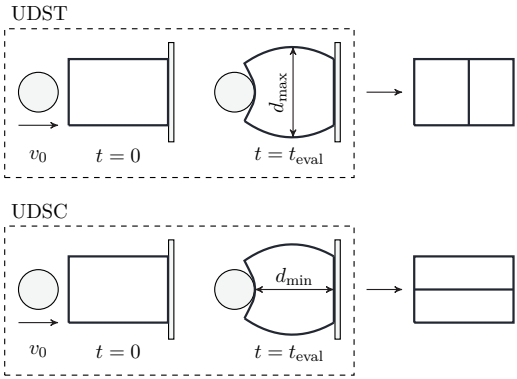


Abbildung 2–12: Funktionsprinzip der Heuristiken UDST und UDSC (modifiziert aus Ortmann et al. (2021))

Die Heuristik *Split Long Edges* (SLE) (deutsch: *Teilen langer Kanten*) soll Wände teilen, die aus langen Kanten abgeleitet sind, um deren Knick- und Beulneigung zu verringern. Die Länge der den Wänden zugehörigen Kanten ist ein wesentlicher Einflussfaktor, wie stark eine Wand dazu neigt zu Knicken oder zu Beulen. Abbildung 2–13 zeigt das Prinzip der Heuristik an einem Beispiel. Die Heuristik sortiert alle Kanten nach ihrer Länge und verbindet die zwei längsten Kanten mittig auf kürzestem Weg. Durch die rein geometrische Funktionsweise der Heuristik benötigt diese keine Simulationsdaten als Grundlage.

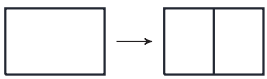


Abbildung 2–13: Funktionsprinzip der Heuristik SLE (modifiziert aus Beyer et al. (2021))

2.5.5 Optimierungsablauf

Der Optimierungsablauf für beliebige Zielfunktionen und Restriktionen der GHT besteht aus zwei ineinander verschachtelten Optimierungsschleifen. Die Aufgabe der äußeren Optimierungsschleife ist es, Topologievorschläge basierend auf den vorgestellten Heuristiken zu generieren. In der inneren Optimierungsschleife werden diese Entwürfe dann entweder mit einem einzelnen Funktionsaufruf simuliert, oder, wenn gefordert, durch eine Dimensionierung weiter optimiert. Der gesamte Optimierungsablauf der GHT ist in Abbildung 2–14 dargestellt.

Zu Beginn wird der initiale Entwurf in Form eines von dem/der Anwender*in definierten Graphen und eines zugehörigen Lastfalls von der GHT eingelesen. Je nach verfügbaren Rechenkapazitäten und Anforderungen an die optimierte Struktur kann die initiale Struk-

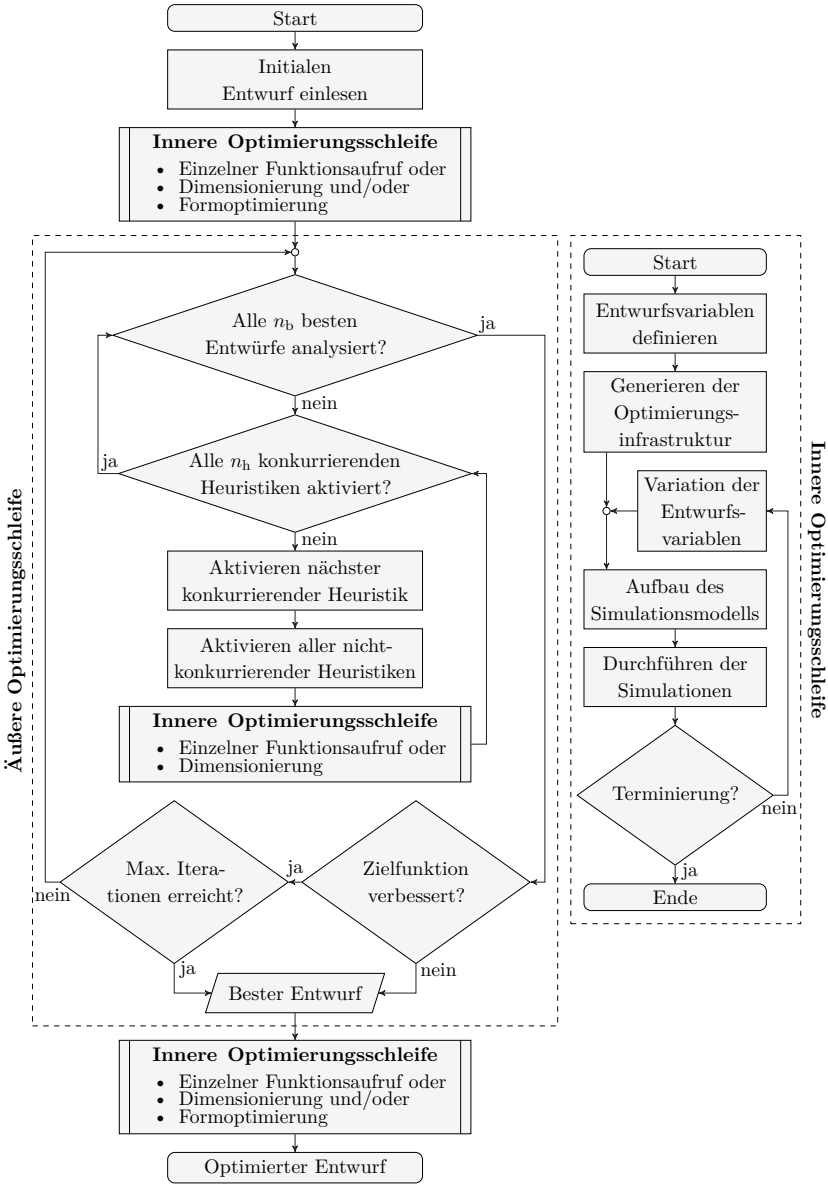


Abbildung 2–14: Optimierungsablauf der GHT (modifiziert aus Beyer et al. (2021))

tur in ihrer Form oder Dimension durch einen Aufruf der inneren Optimierungsschleife zu Beginn des GHT-Prozesses optimiert werden. Im Kontext der GHT wird der Aufruf der inneren Optimierungsschleife mit *Size and Shape Optimization* (SSO) abgekürzt. Dadurch starten die Heuristiken der GHT bereits auf einem guten Entwurf und die GHT wird in eine günstige Richtung gelenkt. In der inneren Optimierungsschleife müssen die Entwurfsvariablen für die vorliegende Optimierungsaufgabe definiert werden. Zusätzlich muss die Infrastruktur der Optimierung generiert werden. Es folgt der grundlegende Ablauf einer Optimierung, wie er bereits in Abbildung 2–2 gezeigt wurde. Der Aufbau des Simulationsmodells geschieht mit GRAMB. Zur Optimierung wird hier typischerweise der *Successive Response-Surface-Model*-Algorithmus (SRSM-Algorithmus) (Stander und Craig 2002) genutzt, welcher explizit für nichtlineare Crashoptimierungen entwickelt wurde. Für den Fall, dass für die Optimierung wenig Ressourcen verfügbar sind oder die Ergebnisse schnell generiert werden müssen, kann die innere Optimierungsschleife auch durch einen einzelnen Funktionsaufruf der initialen Struktur ersetzt werden. Die Durchführung der inneren Optimierungsschleife wird in der GHT mit der Software LS-OPT (Livermore Software Technology Corporation (LSTC) 2023b) umgesetzt.

Anschließend wird die äußere Optimierungsschleife aktiviert, in der die Heuristiken auf die maximal n_b besten Entwürfe der vorigen Iteration angewandt werden. In der ersten Iteration der äußeren Schleife existiert nur der initiale Entwurf. In folgenden Iterationen sind mehr Entwürfe aus vorigen Iterationen verfügbar. Für jeden einzelnen Entwurf der vorigen Iteration wird nun jede der n_h konkurrierenden Heuristiken aktiviert. Für jeden einzelnen neuen Entwurf werden zusätzlich alle nicht-konkurrierenden Heuristiken aktiviert. So werden in einer Iteration maximal $n_\mathrm{b} \cdot n_\mathrm{h}$ Entwürfe generiert und berechnet. Falls ein Entwurf aus den Heuristiken nicht die Fertigungsrestriktionen erfüllt, so wird dieser verworfen und nicht weiterverfolgt. Sofern gewünscht, kann jeder einzelne dieser Entwürfe noch dimensioniert werden, was insbesondere bei Minimierungsaufgaben bezüglich maximaler Kraftpeaks relevant ist, um eine Weichheit der Struktur zu ermöglichen. Theoretisch kann auch jeder Entwurf in seiner Form optimiert werden, was in der Praxis aber nicht gemacht wird, da der Aufwand und der Gewinn an Strukturperformance nicht im Verhältnis stehen.

Die äußere Schleife wird dann verlassen, wenn eines von zwei Abbruchkriterien erfüllt ist. Entweder wurde die Zielfunktion für eine gewisse Anzahl an Iteration nicht verbessert oder die maximale Anzahl an vom Nutzer festgelegten Iterationen wurde erreicht. In beiden Fällen wurde der beste Entwurf der äußeren Optimierungsschleife gefunden. In einem finalen Schritt kann der gefundene Entwurf, wenn gewünscht, noch dimensioniert werden und/oder formoptimiert werden. Als Ergebnis erhält der/die Anwender*in eine mit der GHT optimierte Crashstruktur.

3 Grundlagen des Machine Learnings mit Fokus auf Reinforcement Learning

3.1 Einführung und Einordnung Künstlicher Intelligenz

Künstliche Intelligenz (KI) (englisch: *Artificial Intelligence*) ist ein Werkzeug zur Lösung von komplexen Problemstellungen und gewinnt in der heutigen Welt immer weiter an Bedeutung. Eine einheitliche Definition von KI liegt in der Wissenschaft nicht vor, was sich durch das junge Alter des Forschungsbereichs begründen lässt (Humm et al. 2022). Der Begriff KI wurde 1955 von John McCarthy geprägt; vorher hat Alan Turing (1912–1954) bereits grundlegende Forschungsbeiträge zu KI generiert (Humm et al. 2022). McCarthy et al. (1955) haben KI wie folgt definiert:

Ziel der KI ist es, Maschinen zu entwickeln, die sich verhalten, als verfügten sie über Intelligenz. (von Ertel (2016) übersetzt)

Ertel (2016) äußert Kritik an dieser Definition, da diese den Begriff der Intelligenz nicht einordnet und auch schon einfache Aufgaben, die durch klassische Algorithmen problemlos lösbar wären, inkludiert. Eine von Ertel (2016) favorisierte Definition nach Rich (1983) für KI ist:

Künstliche Intelligenz befasst sich mit der Frage, wie man Computer dazu bringen kann, Dinge zu tun, die Menschen im Moment noch besser können. (übersetzt)

Diese Definition umfasst den Anspruch, dass die KI komplexe Probleme lösen soll, welche Computer mit klassischen Algorithmen nicht lösen können. Zusätzlich stellt sie die Beziehung zwischen Mensch und Computer weiter in den Vordergrund. Der Mensch ist der Maßstab zur Definition von KI (Frochte 2020).

Nach Humm et al. (2022) lassen sich Teilgebiete von KI und den damit geforderten Fähigkeiten nach der o. g. Definition von menschlichen Fähigkeiten ableiten. Die Autoren führen folgende Unterteilung an:

- *Wahrnehmen*: Ein Teilgebiet der KI, welches sich mit der Analyse und Verarbeitung von Bildern und Videos beschäftigt, nennt sich *Computer Vision*.
- *Kommunizieren*: *Natural Language Processing* verfolgt das Ziel die Kommunikation zwischen Mensch und Computer auf Basis einer natürlichen Sprache zu ermöglichen.
- *Lernen*: Zum *Machine Learning* (ML) gehören Methoden, welche es ermöglichen Vorhersagen auf Basis von Modellen zu treffen, wobei die Modelle durch die Verallgemeinerung von Daten lernen.
- *Wissen*: Das Teilgebiet *Knowledge Representation* ermöglicht das Organisieren und Strukturieren komplexer Informationen.
- *Denken*: *Logische Programmierung* ist ein Programmierparadigma, das es ermöglicht durch Angabe von Fakten und Regeln bzgl. einer Problemstellung Anfragen von Nutzer*innen zu beantworten.
- *Handeln*: Im Teilgebiet *Planen* sollen Aktionssequenzen gefunden werden, welche von einem Anfangszustand zu einem Endzustand führen.

KI wird in *schwache* und *starke* KI gegliedert. In die Kategorie schwache KI fallen Anwendungen und Systeme, welche maßgeschneidert auf ein Anwendungsfeld hin entwickelt werden. Solch eine KI ist dann nur für eine ganz spezifische Aufgabe geeignet und lässt sich nicht ohne Modifikationen der KI auf andere Problemstellungen übertragen. Eine starke KI kann sich hingegen über den eigentlichen Anwendungszweck hinaus selbstständig weiterentwickeln (Humm et al. 2022). Nach Frochte (2020) besitzt eine starke KI auch ein Bewusstsein und ein Empfindungsvermögen. Vorteile generiert eine starke KI dann beispielsweise durch das selbstständige Erschließen von Aufgabenbereichen und Problemstellungen und dem Lösen dieser. Während die Entwicklung von schwachen KI-Anwendungen aktueller Forschungsgegenstand ist, sind starke KI-Anwendungen zwar in der Öffentlichkeit präsent, aber nicht in absehbarer Zukunft mit den vorhandenen Methoden und Algorithmen umsetzbar (Frochte 2020; Humm et al. 2022). Alle bisher erforschten Anwendungen und Methoden fallen daher in die Kategorie der schwachen KI.

3.2 Grundlagen des Machine Learnings

Machine Learning (ML) (deutsch: *Maschinelles Lernen*) ist ein Teilgebiet der KI, das sich auf die Entwicklung und Anwendung von Algorithmen konzentriert, die aus Daten Muster und Zusammenhänge lernen. Folgende zwei Definitionen, welche in Géron (2019) angeführt werden, konkretisieren den Begriff weiter. Nach Samuel (1959)[1] lautet eine allgemeine Definition von ML:

> *[Maschinelles Lernen ist ein] Forschungsgebiet, das Computern die Fähigkeit verleiht, zu lernen, ohne explizit programmiert zu werden.* (übersetzt)

Zentral ist bei diesem Zitat, dass primär die Daten den Lernprozess des zugrundeliegenden ML-Modells treiben und nicht ein explizit auf die konkrete Problemstellung angepasstes Programm verwendet wird. Der Begriff *Lernen* hat im ML-Kontext eine präzise Bedeutung. Dafür wird die Definition nach Mitchell (1997) herangezogen:

Ein Computerprogramm lernt aus Erfahrung E in Bezug auf eine Aufgabe T und einem Leistungsmaß P, wenn sich seine Leistung bei T, gemessen durch P, mit der Erfahrung E verbessert. (übersetzt)

3.2.1 Strukturierte und unstrukturierte Daten

Alle ML-Methoden benötigen Daten oder einen Datensatz, also eine Form von gesammelten Daten, auf denen die Algorithmen trainieren oder mit denen sie interagieren können. Der dem Training zugrundeliegende Datensatz besteht aus einer Reihe an Datenpunkten. Dabei beinhalten die einzelnen Datenpunkte eine Repräsentation des anzulernenden Sachverhalts, den sog. *Features*, die aus einer Transformation der Rohdaten gewonnen werden können. Features sind also Merkmale, die den Zustand eines Sachverhalts charakterisieren und durch (Zahlen-)Werte beschrieben werden.

Zur Generierung eines Datensatzes basierend auf verschiedenen Features können Methoden der *Design of Experiments* (DoE) (deutsch: *Statistische Versuchsplanung*) herangezogen werden, welche es erlauben, mit möglichst wenig Aufwand möglichst viele Informationen aus Versuchsreihen zu extrahieren. In dieser Dissertation wird insbesondere das *Latin Hypercube Sampling* (LHS) (McKay et al. 1979) dafür betrachtet.

Wenn die Ansammlung an Features über alle Datenpunkte hinweg tabellarisch aufgefasst werden kann, spricht man von strukturierten Daten. Die Struktur eines solchen Datensatzes wird in Tabelle 3–1 aufgezeigt. Für alle m Datenpunkte liegt je eine Konfiguration an Werten der Features f_1 bis f_n vor.

[1] Das Zitat wird in der Praxis häufig der o. g. Veröffentlichung zugeschrieben, obwohl es dort nicht explizit ausformuliert steht

Tabelle 3–1: Aufbau eines strukturierten Datensatzes aus Features f_i und m Datenpunkten (modifiziert aus Frochte (2020))

	f_1	f_2	f_3	$\cdots$	f_{n-2}	f_{n-1}	f_n
1							
2							
$\vdots$							
$m-1$							
m							

Neben den strukturierten Daten können Daten auch unstrukturiert vorliegen, wie es beispielsweise bei Graphen der Fall ist. Jedem Vertex und jeder Kante können Features zugewiesen werden. Die Graphen können sich in der Anzahl an Vertices und Kanten unterscheiden, weshalb die Daten nicht ohne weiteres tabellarisch dargestellt werden können. In Abbildung 3–1 wird ein Beispielgraph dargestellt.

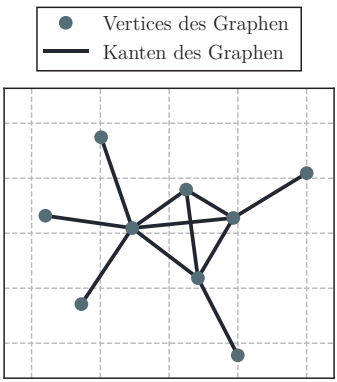


Abbildung 3–1: Darstellung eines Beispielgraphen

3.2.2 Lernparadigmen des Machine Learnings

ML-Algorithmen lassen sich in drei zentrale Lernparadigmen unterteilen. Für jede dieser Klassen gibt es typische Problemstellungen, welche in Abbildung 3–2 dargestellt werden. Neben der klassischen Unterteilung in *Supervised Learning* (SL) (deutsch: *Überwachtes Lernen*), *Unsupervised Learning* (UL) (deutsch: *Unüberwachtes Lernen*) und *Reinforcement Learning* (RL) (deutsch: *Bestärkendes Lernen*) gibt es zusätzlich das *Semi-Supervised Learning*, einer Mischform der beiden Lernparadigmen SL und UL (Géron 2019). Auf diese wird im Rahmen dieser Arbeit nicht weiter eingegangen.

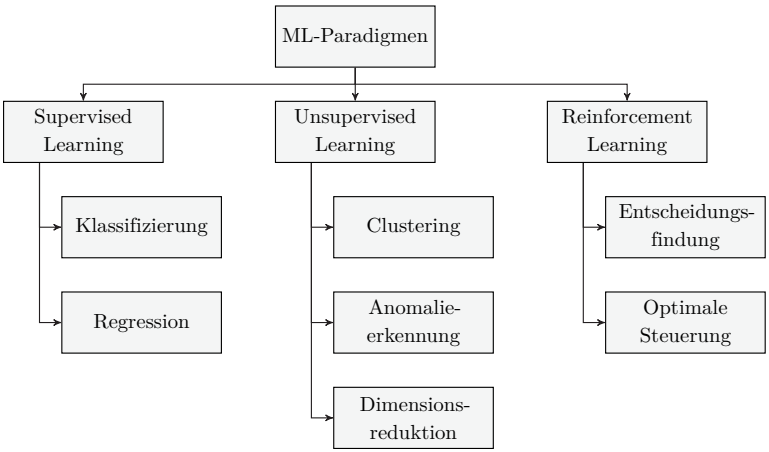


Abbildung 3–2: Übersicht und Einordnung von ML-Lernparadigmen und typischen ML-Problemstellungen

Supervised Learning: Zum SL gehören Methoden, die i. Allg. Klassifikations- und Regressionsproblemstellungen lösen. Sie zeichnen sich dadurch aus, dass sie in Input-Output-Paaren vorliegen und eine hinreichend große Menge an Daten vorhanden ist, die von den Algorithmen als Lernbasis genutzt werden, um ein ML-Modell zu trainieren (Frochte 2020).

Die Datenpunkte enthalten beim SL neben den Features und deren Werten auch Labels. Labels sind den Datenpunkten eines Datensatzes zugeordnet und beschreiben eine Zielgröße, die in der jeweiligen Problemstellung von Interesse ist. Sie enthalten die wahren Werte der Zielgröße und treiben den Lernprozess im SL. Anschaulich bedeutet das, dass beispielsweise in Tabelle 3–1 eine oder mehrere neue Spalten mit Labels hinzugefügt werden, welche den Werten der Features f_1 bis f_n zugeordnet sind und die vorherzusagende Zielgröße korrekt angeben.

So lässt sich durch den Vergleich des Labels mit der Vorhersage des ML-Modells zusammen mit einer Fehlerfunktion die Performance des Modells bestimmen. Um die bestmögliche Performance eines ML-Modells zu erreichen, wird diese Fehlerfunktion durch Anpassung von Modellparametern minimiert.

Bei einem *Klassifikationsproblem* sind die Labels diskret. Die Labels geben die Klasse der zugehörigen Konfiguration an Features an. Ein gängiger Benchmark für ein Klassifikationsproblem ist das Erkennen handgeschriebener Ziffern aus der *Modified National Institute of Standards and Technology*-Datenbank (MNIST-Datenbank) (LeCun et al. 1998).

Algorithmen zum Lösen von Klassifikationsproblemen gibt es zahlreich, wie beispielsweise *Bayes-Klassifikatoren* (Frochte 2020), *k-Nearest Neighbors* (kNNs) (Cover und Hart 1967) und *Support Vector Machines* (SVMs) (Boser et al. 1992; Cortes und Vapnik 1995). Auf diese wird hier nicht weiter eingegangen, da sie zum Verständnis der Dissertation nicht essentiell sind. Stattdessen werden einige Algorithmen knapp erläutert, die zumindest peripher in dieser Dissertation zum Einsatz kommen. Algorithmen, die intensiv genutzt werden, werden in späteren Abschnitten ausführlicher besprochen.

Zu den genutzten Algorithmen zählen:

- *Decision Trees* (DTs) (Breiman et al. 1984; Quinlan 1986) – Der Algorithmus baut eine Baumstruktur (Ernst et al. 2020) auf. Der Baum wird von der Wurzel bis hin zu den Blättern durchschritten. Die inneren Knoten des Baumes formulieren Kriterien an die Features. Je nach Erfüllung des jeweiligen Kriteriums, wird ein anderer Zweig des Baums weiterverfolgt, bis die Blätter des Baumes erreicht werden. Die Blätter beinhalten dann die gesuchte Klasse zu den zugehörigen Features. Der Aufbau des Baumes und die Bestimmung der Kriterien wird durch Kennzahlen, wie beispielsweise der Entropie, gesteuert. Entropie beschreibt in dem Kontext der Informationstheorie den mittleren Informationsgehalt von Daten (Shannon 1948; Ernst et al. 2020). Je höher die Entropie ist, desto weniger strukturiert sind die Daten und desto mehr Informationen werden benötigt, die Daten zu repräsentieren. Entsprechend lassen sich weniger Informationen aus den Daten ableiten. Ein Entscheidungsbaum wird also so aufgebaut, dass die Entropie der Zielklasse minimiert wird.
- *Random Forests* (RFs) (Breiman 2001) – Bei einem RF werden viele unterschiedliche DTs trainiert und die Modelle für die Klassenvorhersage kombiniert. So lässt sich die Varianz der einzelnen Modelle in den Vorhersagen verringern. Ein solches Vorgehen nennt man i. Allg. *Bootstrap Aggregation* oder auch *Bagging* (Breiman 1996).
- *Neuronale Netze* (NNs) (McCulloch und Pitts 1943; Rosenblatt 1958) – Ein konventionelles NN besteht aus Neuronen, die in sequentiellen Schichten angeordnet sind. Die Verbindungen der Neuronen aus benachbarten Schichten sind parametrisiert. Schicht für Schicht werden die Features durch das Netz propagiert. Die Ausgabe des NNs ist dann beispielsweise eine Wahrscheinlichkeitsverteilung über die Klassen der jeweiligen Problemstellung. Trainiert werden moderne NNs häufig mit einem effizienten, *stochastischen Gradientenverfahren* (Robbins und Monro 1951). Dabei handelt es sich um eine stochastische Variante klassischer Gradientenverfahren, bei dem eine zufällig ausgewählte Teilmenge des Datensatzes zur Gradientenbestimmung herangezogen wird. Dadurch ist der Rechenaufwand bei großen, hochdimensionalen Datensätzen geringer. Bestimmt werden die Gradienten basierend auf dem sog. *Backpropagation-Algorithmus* (Rumelhart et al. 1986) (deutsch: *Fehlerrückführungsalgorithmus*).

Bei einem *Regressionsproblem* sind die Labels typischerweise kontinuierlich. Für die Features in jedem Datenpunkt ist ein Label zugeordnet. Dieser funktionale Zusammenhang soll im Trainingsprozess angelernt werden. Damit sind Methoden des MLs zum Lösen von Regressionsproblemen artverwandt mit Response-Surface- und Metamodellen, wie sie in Schumacher (2020) beschrieben werden. Prinzipiell unterscheidet sich damit das Regressionsproblem und das Klassifikationsproblem nur durch die Art und Weise der Definition der Labels. Aufgrund der Ähnlichkeit zum Lösen des Klassifizierungsproblems lassen sich viele der o. g. Algorithmen auch zum Lösen von Regressionsproblemstellungen heranziehen.

Unsupervised Learning: Problemstellungen, die sich dadurch auszeichnen, dass keine Labels zum Trainieren nötig bzw. vorhanden sind, gehören zum UL. Ein grundlegendes Ziel ist dabei, intrinsische Muster und Strukturen innerhalb des Datensatzes zu finden (Frochte 2020). Da keine Labels bekannt bzw. vorhanden sind, kann hier keine Fehlerfunktion bezüglich einer Zielgröße definiert werden.

Das Clustering von Daten und die Anomalieerkennung sind artverwandte Problemstellungen des ULs. Beim Clustering wird der Datensatz in Cluster bzw. Gruppen eingeteilt. Datenpunkte mit geringerer Distanz und damit mit ähnlichen Features sind Bestandteil einer Gruppe. Der Unterschied zur Klassifikation besteht darin, dass keine Labels vorhanden sind. Somit ist zu Beginn des Trainings unklar, welche Gruppen sich ausbilden werden. Für neue Datenpunkte kann jetzt überprüft werden, zu welcher Gruppe sich die neuen Datenpunkte zuordnen lassen bzw. welche Gruppe den neuen Daten am nächsten ist. Bei der Anomalieerkennung wird im Gegensatz zum Clustering überprüft, ob es sich um statistische Ausreißer handelt. Ein solcher Datenpunkt ist dann weit entfernt von den Gruppen, die sich beim Clustering gebildet haben. Als prominentes Beispiel für einen Algorithmus zum Clustern von Datensätzen ist exemplarisch der *k-Means-Algorithmus* (MacQueen 1967) zu nennen.

UL kann auch zur Dimensionsreduktion verwendet werden. Dimensionsreduktionsalgorithmen eignen sich, um komplexe Featureräume zu vereinfachen. Dazu wird eine Einbettung der Features bestimmt, die üblicherweise durch einen Vektor repräsentiert wird. Diese Einbettung charakterisiert den zugehörigen Datenpunkt bestmöglich, liegt aber in einer niedrigdimensionalen Darstellung als der originale Featureraum vor. Der Informationsgehalt zwischen den originalen Features und der Einbettung soll möglichst gleich groß sein. Je kompakter die Einbettung gewählt wird, desto schwieriger ist es i. Allg. den Informationsverlust gering zu halten. Redundante, stark korrelierende Features werden so automatisch aus den Daten herausgerechnet. Dadurch, dass sich die Dimensionalität des Featureraums mit geringem Informationsverlust verkleinern lässt, eignen sich Dimensionsreduktionsalgorithmen, um den Featureraum für den *Downstream-Task* aufzubereiten und dessen Training zu vereinfachen. Der Downstream-Task ist dabei die eigentlich zu lösende Aufgabe, welche hier durch das Preprocessing der Daten erleichtert werden soll. Typische Algorithmen der Dimensionsreduktion sind exemplarisch die lineare

Principal Component Analysis (PCA) (Pearson 1901) und der nichtlineare *t-distributed Stochastic Neighbor Embedding*-Algorithmus (t-SNE-Algorithmus) (van der Maaten und Hinton 2008).

Reinforcement Learning: Die Kernidee des RLs ist, dass mittels *Trial-and-Error* eine Strategie angelernt wird, die in einer Umgebung bestmöglich im Sinne einer numerisch konkret definierten Aufgabe handelt. Die Umgebung ist dabei möglicherweise stochastisch und nicht-stationär. Da das RL eine besonders hohe Relevanz für diese Dissertation hat, wird dieses separat beschrieben.

3.2.3 Feature-Engineering

In praktischen Aufgabenstellungen, die mit ML zu lösen sind, gibt es viele Features, die nicht direkt für ein Training eines ML-Modells geeignet sind und erst aufbereitet werden müssen. Das systematische Vorgehen zum Aufbereiten des Datensatzes für ein effizientes Training eines ML-Modells nennt sich Feature-Engineering. So passiert es in realen Datensätzen beispielsweise häufig, dass Features nicht besetzt sind, d. h., dass keine Werte für einen Teil des Datensatzes verfügbar sind. Dann müssen Annahmen oder Abschätzungen getroffen werden, welche Werte stellvertretend angegeben werden, um das Eingangsformat der Daten, die das ML-Modell erwartet, aufrecht zu erhalten. Dieses Vorgehen nennt sich Imputation.

Um den Trainingsprozess zu vereinfachen, sollten nach Frochte (2020) nicht zwingend alle vorhandenen Features genutzt werden. Stattdessen sollte eine fundierte und überlegte Auswahl an Features getroffen werden. Das bedeutet beispielweise, dass stark korrelierende Features aussortiert werden sollten, da diese keinen inhaltlichen Mehrwert zum Training beisteuern können und somit den Trainingsprozess aufgrund der höheren Anzahl an Eingangsgrößen erschweren. Ebenso könnte ein Feature keine Korrelation bzgl. einer konkreten Zielgröße haben. Dann ist es sinnvoll das Feature zu streichen und gar nicht erst im Trainingsprozess zu verwenden. Für hochdimensionale Problemstellungen kann es sich anbieten, die Dimensionalität durch Dimensionsreduktionsmethoden zu reduzieren.

Ob und wie stark ein Feature für den Trainingsprozess geeignet ist, kann quantifiziert werden. In dieser Arbeit geschieht das auf Basis der *Permutation Feature Importance* (PFI) (Breiman 2001). Die PFI gibt an, wie wichtig die konkrete Wahl des Features ist. Dabei ist es irrelevant, ob das Feature aktiv oder inaktiv für ein ML-Modell ist. Entscheidend ist lediglich, dass das Feature wichtig ist, um auf die Performance des Modells schließen zu können.

Zur Bestimmung der PFI wird ein Regressionsmodell aufgebaut, welches häufig die Modellperformance in Abhängigkeit der Features abbildet. Das Bestimmtheitsmaß R^2 (Wright 1921) ist ein Konzept aus der Statistik und gibt die Anpassungsgüte eines solchen

Regressionsmodells bezüglich eines Datensatzes an. Typischerweise gilt $0 \leq R^2 \leq 1$, wobei der Idealfall $R^2 = 1$ ist. Das Regressionsmodell ist dann so gut, dass es alle Datenpunkte des Datensatzes interpoliert. Ein $R^2 = 0$ bedeutet, dass das Regressionsmodell den Datensatz nicht gut abbildet. Theoretisch kann auch ein $R^2 < 0$ auftreten. Das passiert dann, wenn das Regressionsmodell schlechter ist als ein Modell, welches stets den Mittelwert der Labels wählt. Deshalb ist es in solchen Ausnahmefällen auch möglich, dass die PFI negativ ist. Für die Berechnung des Bestimmtheitsmaßes gilt basierend auf den wahren Funktionswerten y_i, der modellbasierten Vorhersage $\tilde{y}$ und dem Mittelwert aller Funktionswerte $\overline{y}$:

$$R^2 = 1 - \frac{\sum_i (y_i - \tilde{y}_i)^2}{\sum_i (y_i - \overline{y})^2}, \tag{3–1}$$

Im Anschluss wird zur Bestimmung der PFI jedes Feature einzeln zufällig permutiert, wodurch die Abhängigkeit zwischen dem betrachteten Feature und der Performance des Modells verloren geht. Die Differenz zwischen dem Bestimmtheitsmaß R^2 mit und ohne permutiertem Feature gibt Auskunft darüber, wie wichtig dieses Feature zur Vorhersage der Zielgröße ist.

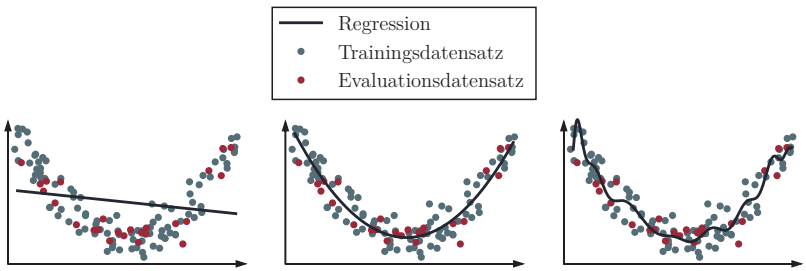
Zusätzlich kann zur Quantifizierung der Eignung von Features beispielsweise der Rangkorrelationskoeffizient nach Spearman (1904) betrachtet werden. Dieser gibt an, wie gut sich die Abhängigkeit zwischen zwei Variablen durch eine monoton steigende oder fallende Funktion abbilden lässt. Hier sind das zum einen beispielsweise das betrachtete Feature und zum anderen welche Modellperformance mit diesem Feature erreicht wird. Sinkt die Performance tendentiell bei Nutzung des Features, ist die Korrelation negativ. Steigt die Performance bei Nutzung des Features, ist sie hingegen positiv. Durch Betrachtung der Monotonie kann der Rangkorrelationskoeffizient nichtlineare Effekte bemessen. Das ermöglicht Auskunft darüber, ob ein Feature genutzt werden sollte.

3.2.4 Modellauswahl

Das Ziel beim Trainieren eines ML-Modells ist es, dass dieses über die zum Training herangezogenen Daten hinaus sinnvolle Vorhersagen trifft. Man spricht hier von der *Generalisierung* des Datensatzes. Das ist entscheidend, denn nicht der Selbstzweck ist das Ziel des Trainings von ML-Modellen, sondern die eigentliche zu lösende Aufgabe, der Downstream-Task.

Wird ein ML-Modell auf einem einzelnen Datensatz trainiert, kann es passieren, dass *Under-* bzw. *Overfitting* auftritt. Beim Overfitting lernt das Modell, den zum Trainieren herangezogenen Datensatz, den Trainingsdatensatz, „auswendig“. Dabei passt sich das Modell zu sehr dem Trainingsdatensatz an und bildet Effekte wie Rauschen in den Daten mit ab. Darunter leidet die Verallgemeinerungsfähigkeit auf neuen, ungesehenen Daten. Da beim Overfitting die Performance auf den Trainingsdaten sehr gut ist, ist Overfitting

schwierig zu detektieren. Underfitting bedeutet, dass das Modell nicht anpassungsfähig genug ist, um die Problemstellung in ihrer grundlegenden Charakteristik anzulernen. Underfitting ist allerdings weniger kritisch als Overfitting, da sich Underfitting in der Performance des Modells während des Trainings auch auf den Trainingsdaten bemerkbar macht. Abbildung 3–3 visualisiert das Konzept des Under- und Overfittings anhand eines Regressionsmodells aus dem SL, in dem zu erkennen ist, dass die Komplexität des Modells im Falle des Underfittings zu klein und im Falle des Overfittings zu groß ist.



(a) Underfitting **(b)** Optimaler Fit **(c)** Overfitting

Abbildung 3–3: Under- und Overfitting im Vergleich zu einem optimalen Fit einer Regression durch einen beliebigen Datensatz

Es gibt viele verschiedene Möglichkeiten Overfitting während des Trainings eines ML-Modells zu detektieren. Beispielsweise kann die beim Training herangezogene Fehlerfunktion genutzt werden und auf einem zusätzlichen Evaluationsdatensatz evaluiert werden. Dieser Evaluationsdatensatz beinhaltet neue, für das ML-Modell ungesehene Daten, welche analog zum Trainingsdatensatz generiert werden. Wenn beim Training die Fehlerfunktionswerte der Trainingsdaten immer weiter sinken und die Fehlerfunktionswerte der Evaluationsdaten stagnieren, hat das Modell Probleme mit der Verallgemeinerung der Trainingsdaten.

Zwar kann mit dieser Herangehensweise überprüft werden, ob das Modell den Trainingsdatensatz overfittet, doch bleibt eine implizite Abhängigkeit zu den Evaluationsdaten zur Bewertung der Modellperformance bestehen. Dies tritt dann merklich auf, wenn die Evaluationsdaten besonders günstig oder ungünstig verteilt sind und damit beim Training eine zu gute oder zu schlechte Performance des ML-Modells vorgetäuscht wird.

Um diese Abhängigkeit weiter zu reduzieren, wird in dieser Dissertation, sofern praktisch möglich, das Prinzip der k-Fold-Cross-Validation genutzt. Eine frühe Beschreibung dieser Methode findet sich in den Ausführungen von Stone (1974). Anstelle eines fixen Evaluationsdatensatzes wird hier der Trainingsdatensatz in k Folds aufgeteilt. Ein Fold bezeichnet jede von k gleichgroßen Teilmengen des Trainingsdatensatzes, die iterativ über k Splits zum Evaluieren verwendet werden, während die verbleibenden $k-1$ Teilmengen zum Trainieren verwendet werden. Die Aufteilung der Folds rotiert solange, bis jeder

Fold einmal als Evaluationsfold genutzt wurde. Auf Basis des Evaluationsfolds wird ein Overfitting des Modells detektiert. Das Modell wird folglich k-mal neu trainiert und dessen Performances am Ende zu einem vom Datensatz statistisch unabhängigeren Mittelwert zusammengefasst. Abbildung 3–4 zeigt das Prinzip der k-Fold-Cross-Validation.

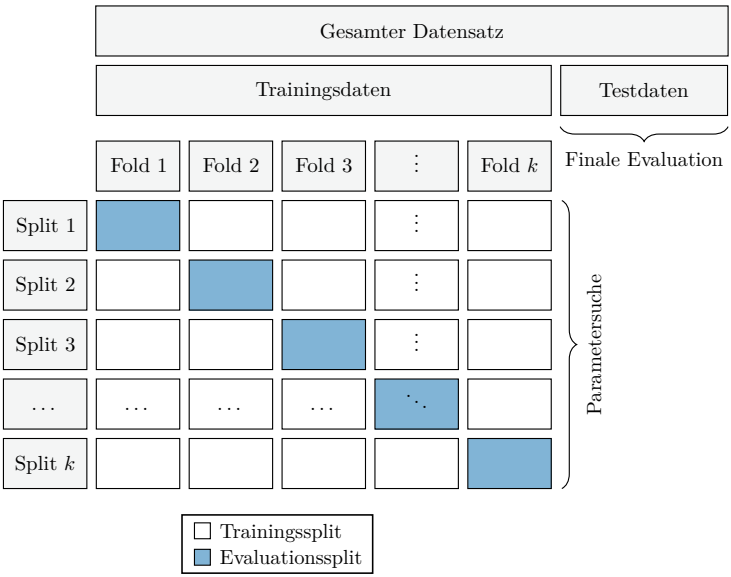


Abbildung 3–4: Prinzip der k-Fold-Cross-Validation (modifiziert aus Pedregosa et al. (2011))

Das Training des Modells über die k Folds ist aufgrund der robusten Schätzung der Performance geeignet, um die optimalen Parameter des Modells zu bestimmen. Für eine finale Bestimmung der Performance des Modells muss dann aber ein weiterer, für das Modell ungesehener Testdatensatz herangezogen werden. Dadurch wird vermieden, dass die final ermittelte Performance des Agenten abhängig von dem gesplitteten Trainingsdatensatz ist. Wichtig ist, dass das Modell und seine Parameter nach der finalen Evaluation auf den Testdaten nicht mehr verändert werden darf. Würde jetzt weiter an der Performance des Modells gearbeitet werden, damit das Modell auch auf dem Testdatensatz besonders gut performt, dann ist der Schätzer der Modellperformance wieder beeinflusst und nicht mehr repräsentativ. Man spricht dann von einem *Data Leakage* (deutsch: *Datenleck*).

3.3 Neuronale Netze

Bei einem *Neuronalen Netz* (NN) handelt es sich um ein Netzwerk von miteinander verschachtelten Nervenzellen, die es Menschen und Tieren ermöglichen motorische und intellektuelle Fähigkeiten zu erlernen (Ertel 2016). Diese biologische Vorlage wird für ein abstraktes Modell herangezogen, welches dann Problemstellungen im Bereich des ML erlernen und lösen kann. Häufig wird dieses abstrakte Modell zur Differenzierung von der biologischen Vorlage auch *künstliches* NN genannt. Im Rahmen dieser Dissertation wird der Begriff *Neuronales Netz* gebraucht und meint stets das mathematische Modell.

Die grundlegenden Konzepte von NNs führen auf McCulloch und Pitts (1943) zurück. Dort wurde ein simples Neuronenmodell in Analogie zur biologischen Grundlage vorgeschlagen, welches heutzutage in abgewandelter Form auch in NNs genutzt wird. Das von Rosenblatt (1958) entwickelte Perzeptron gilt als das erste NN und stellt die Basis für heutige NNs dar. Es handelt sich bei einem Perzeptron um die fundamentale Komponente eines NNs. Das Perzeptron empfängt eine Reihe von Eingaben, multipliziert jede Eingabe mit einem Gewicht, summiert sie und übergibt das Ergebnis an eine Stufenfunktion, um eine Ausgabe zu erzeugen. Ein weit verbreiteter Ansatz eines NNs ist die Verschachtelung von Perzeptrons zu *Multilayer Perceptron* (MLP).

Heutige NNs sind vielfältig und bestehen aus unterschiedlichen Strukturen und Architekturen. Für diese Dissertation sind zusätzlich *Convolutional Neural Networks* (CNNs) wichtig, welche sich insbesondere für räumliche Daten wie Bilder eignen. Eine weitere gängige Architektur ist das *Long Short-Term Memory* (LSTMs) (deutsch: *Langes Kurzzeitgedächtnis*) (Hochreiter und Schmidhuber 1997). Diese eignet sich zur Verarbeitung von Zeitsignalen. Für Graphdaten eignen sich *Graph Neural Networks* (GNNs) (Gori et al. 2005; Scarselli et al. 2009) und *Graph Convolutional Networks* (GCNs) (Kipf und Welling 2017), welche beliebige Graphen einlesen können, unabhängig ihrer Vertex- und Kantenanzahl. Dabei sind GCNs eine Form von speziellen GNNs, welche ähnlich zu den CNNs auf Graph-Convolutions basieren.

3.3.1 Multilayer Perceptron

Das *Multilayer Perceptron* (MLP) (deutsch: *Mehrlagiges Perzeptron*) ist eine Verschachtelung von einzelnen Perzeptrons, wie sie von Rosenblatt (1958) vorgestellt werden. Die Verwendung des Begriffs MLP ist ambig und kann je nach Kontext unterschiedlich aufgegriffen werden. In dieser Arbeit wird unter einem MLP ein Feed-Forward-Netzwerk mit einem Input-Layer, einem oder mehreren Hidden-Layer und einem Output-Layer verstanden. Feed-Forward-Netze zeichnen sich dadurch aus, dass die Eingabedaten vom Input-Layer sequentiell, also von Layer zu Layer, bis zum Output-Layer propagiert werden. Eine schematische Darstellung eines MLP wird in Abbildung 3–5 dargestellt, bei der die Neuronen durch Kreise repräsentiert werden.

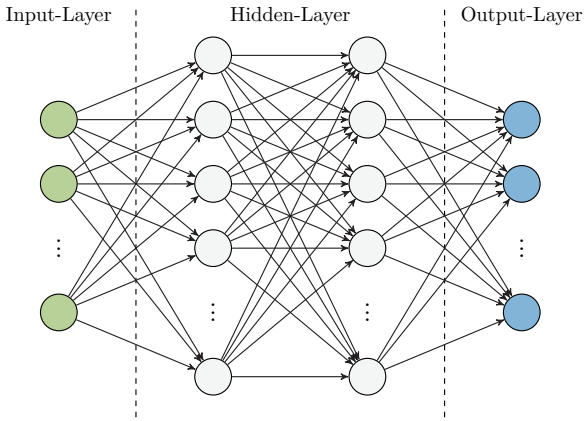


Abbildung 3–5: Beispiel eines MLPs mit einem Input-Layer, zwei Hidden-Layern und einem Output-Layer

Die Anzahl der Neuronen der Input- und Output-Layer sind typischerweise durch die Problemstellung vorgegeben. So ist die Anzahl an Neuronen im Input-Layer durch die Anzahl der betrachteten Features bestimmt. Jedes Neuron im Input-Layer hält somit einen dem Feature zugehörigen Wert. Die Neuronenanzahl im Output-Layer ist durch die Dimensionalität der Zielgröße bestimmt, beispielsweise der Anzahl an vorhandenen Klassen in einem Klassifizierungsproblem oder der Anzahl an Ausgabewerten bei einem Regressionsproblem. Die Anzahl an Hidden-Layer und die Anzahl der zugehörigen Neuronen ist üblicherweise von dem/der Nutzer*in festzulegen. Je mehr Hidden-Layer und Neuronen das NN aufweist, desto komplexere Problemstellungen können mit diesem gelöst werden.

Wie genau das Propagieren der Informationen in einem MLP funktioniert, wird mithilfe von Abbildung 3–6 erklärt. Dargestellt wird die Verarbeitung von Informationen für ein einzelnes Neuron. Auf jedes Neuron aller Layer, mit Ausnahme des Input-Layers, wird das dargestellte Vorgehen angewandt. Die Inputs in das Neuron entsprechen den Werten der Neuronen, den sog. Aktivierungen, aus dem jeweils vorigen Layer. Diese Inputs werden auf Basis einer Propagierungs- bzw. Übertragungsfunktion verrechnet. Typischerweise ist die Propagierungsfunktion eine mit Gewichten w_{ij} gewichtete Summe der Inputs x_i. Zu dem Ergebnis dieser Linearkombination wird ein Bias dazu addiert. Der Bias des Neurons ist ein Skalar, der das Aktivierungsniveau des Neurons steuert. Die Summe aus gewichteten Inputs und dem Bias wird an eine Aktivierungsfunktion φ übergeben. Diese Aktivierungsfunktion spielt eine zentrale Rolle bei der Funktionsweise von NNs, denn diese ermöglicht die Approximation komplexer, nichtlinearer Funktionen. Der Bias hat die Aufgabe den Wirkbereich der Aktivierungsfunktion zu verschieben. Die Aktivierung

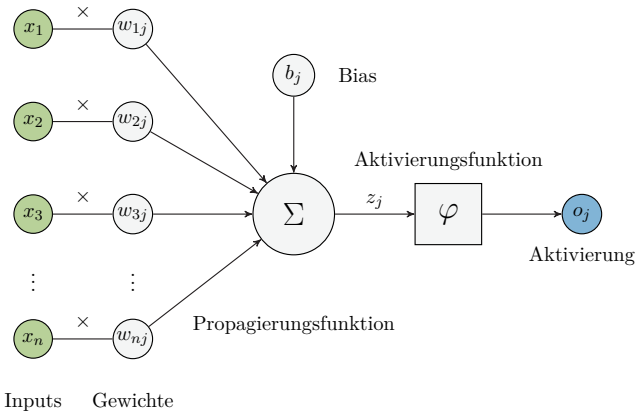


Abbildung 3–6: Aufbau eines Neurons in einem MLP

o_j des j-ten Neurons eines Layers mit n eingehenden Inputs des vorigen Layers i lässt sich mit dem Ergebnis z_j aus der Propagierungsfunktion final zu

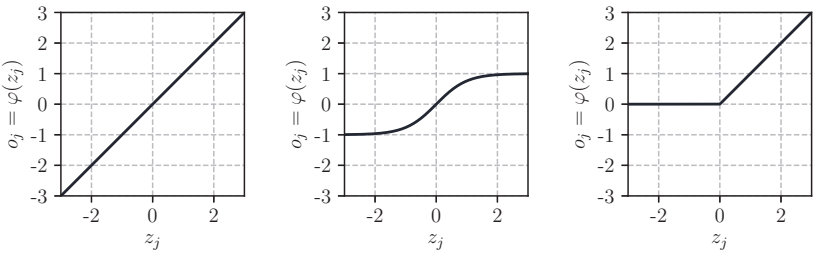
$$o_j = \varphi(z_j) \quad \text{mit} \quad z_j = \sum_i^n w_{ij} x_i + b_j \tag{3–2}$$

bestimmen.

In Abbildung 3–7 wird eine Auswahl an einfachen und häufig genutzten Aktivierungsfunktionen aufgezeigt. Die für die Problemstellung besten Aktivierungsfunktionen sind a priori nicht bekannt und müssen durch eine systematische Suche bestimmt werden. Theoretisch könnte jedem Neuron innerhalb eines Layers eine andere Art von Aktivierungsfunktion zugeordnet sein. Das wird so in der Praxis für klassische NN nicht gemacht. Standardmäßig erhalten alle Neuronen innerhalb eines Layers dieselbe Aktivierungsfunktion. Dadurch wird die Komplexität der Problemstellung reduziert.

Die Identität eignet sich beispielsweise für den Output-Layer bei Regressionsproblemen, denn so lässt sich der Wertebereich der Zielfunktion vollständig abdecken. Für den Fall, dass die Aktivierungsfunktion *aller* Layer bzw. Neuronen der Identität entspricht, handelt es sich bei den Aktivierungen der Neuronen stets um Linearkombinationen der Inputs aus dem Input-Layer unabhängig der Anzahl der Hidden-Layer und der Neuronen je Layer. Ein solches Vorgehen ist also i. d. R. nicht sinnvoll, da nur lineare Approximationen gefunden werden können.

Die Verwendung anderer Aktivierungsfunktionen ermöglicht das Einbringen von Nichtlinearitäten in das NN, wodurch dieses auch komplexe Zusammenhänge anlernen kann. Beispiele für häufig genutzte Aktivierungsfunktionen sind der *Tangens Hyperbolicus* (Tanh) und die *Rectified Linear Unit*-Funktion (ReLU-Funktion). Beide bilden einen



(a) Identität **(b)** *Tangens Hyperbolicus* (Tanh) **(c)** *Rectified Linear Unit* (ReLU)

Abbildung 3–7: Drei Beispiele für gängige Aktivierungsfunktionen

unterschiedlichen Wertebereich ab und reagieren unterschiedlich auf verschiedene Eingangswerte. Welche Aktivierungsfunktion für eine konkrete Aufgabenstellung die beste ist, ist nicht zwingend eindeutig und muss in einer entsprechenden Untersuchung herausgefunden werden. Dabei haben die Aktivierungsfunktionen nicht nur Einfluss auf die Komplexität und den Aufbau des NNs, sondern können bei geschickter Wahl auch den Trainingsprozess im Allgemeinen vereinfachen.

MLPs lassen sich also durch unterschiedlichste Parameter beschreiben. Unterschieden wird zwischen den trainierbaren Parametern und den nicht-trainierbaren Hyperparametern. Die trainierbaren Parameter sind üblicherweise

- die Gewichte der Verknüpfungen zwischen Neuronen und
- der Bias der Neuronen.

Trainiert werden die Parameter häufig mit der bereits zuvor benannten Kombination aus dem Backpropagation-Algorithmus (Rumelhart et al. 1986), der die Gradienten zur Anpassung der Modellparameter im Sinne einer Fehlerfunktion findet und einem stochastischen Gradientenverfahren (Robbins und Monro 1951), der die Parameter entsprechend anpasst.

Zu den vom Nutzer festzulegenden Hyperparametern gehören meist

- die Anzahl der Hidden-Layer,
- die Anzahl der Neuronen je Hidden-Layer und
- die Aktivierungsfunktion je Layer.

Eine Einschätzung der Wahl der besten Hyperparameter ist a priori nur eingeschränkt möglich. Um die bestmögliche Performance des MLPs zu ermitteln, wird deshalb i. d. R. eine systematische Suche nach den am geeignetsten Hyperparametern durchgeführt. Hier eignet sich exemplarisch die in Abschnitt 3.2.4 vorgestellte k-Fold-Cross-Validation.

3.3.2 Convolutional Neural Network

Bei einem *Convolutional Neural Network* (CNN) (LeCun et al. 1998) handelt es sich um eine Art von NN, welches besonders gut mit räumlichen bzw. tabellarischen Daten mit Nachbarschaftsbeziehung, wie es beispielsweise bei Bildern der Fall ist, umgehen kann. Die folgende Beschreibung orientiert sich an den Ausführungen von Frochte (2020). Die Methodik beschränkt sich nicht nur auf zweidimensionale Strukturen wie Bilder. Auch eindimensionale Zeitreihen oder dreidimensionale, räumliche und in Voxel unterteilte Strukturen können mit CNNs sinnvoll angelernt werden, solange eine Nachbarschaftsbeziehung in den Daten gegeben ist. Der prinzipielle Aufbau eines CNNs wird in Abbildung 3–8 aufgezeigt und die Bestandteile werden im Folgenden erklärt.

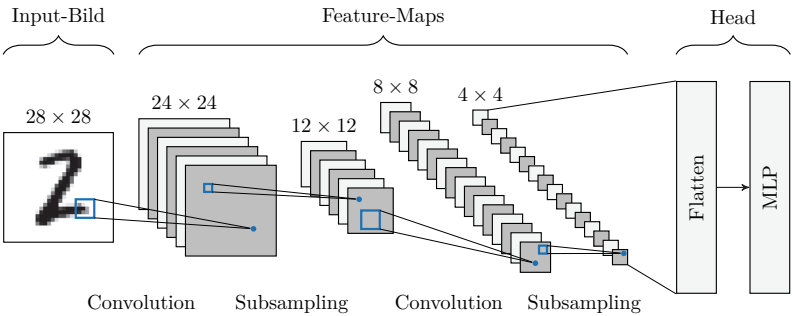


Abbildung 3–8: Prinzipieller Aufbau und Funktionsweise eines CNNs am Beispiel einer Klassifizierung handgeschriebener Ziffern (modifiziert aus LeCun et al. (1998))

Convolution: Ein zentraler Bestandteil eines CNNs ist die *Convolution* (deutsch: *Faltung*), welche sich aus dem Input $\boldsymbol{I}$ des betrachteten Layers des CNNs und dem Kernel $\boldsymbol{K}$, welcher auch Filter genannt wird, zusammensetzt. Der Input $\boldsymbol{I}$ ist üblicherweise das Eingangsbild oder in folgenden Layern des CNNs eine Feature-Map. Es handelt sich um eine Matrix, dessen Einträge die anzulernenden Parameter des CNNs darstellen. Ähnlich zum Training der MLPs werden die Gradienten zur Anpassung der Parameter im Kernel über den Backpropagation-Algorithmus (Rumelhart et al. 1986) bestimmt. Der Kernel definiert, welche Features aus dem Input extrahiert werden. Damit kann der Kernel beispielsweise so antrainiert werden, dass dieser vertikale oder horizontale Kanten im Input erkennt, was für den Downstream-Task wichtig sein könnte. Die Dimension des Kernels ist ein Hyperparameter und ist vom Anwender festzulegen.

Für den für diese Arbeit relevanten diskreten Fall in zwei Dimensionen zur Verarbeitung von Bildern ist die Convolution über

$$\boldsymbol{S}[i,j] = (\boldsymbol{I} * \boldsymbol{K})[i,j] = \sum_m \sum_n \boldsymbol{I}[m,n] \cdot \boldsymbol{K}[i-m, j-n] \tag{3–3}$$

definiert. Das Ergebnis $\boldsymbol{S}[i,j]$ der Faltung ist die sog. Feature-Map in den Zeilen und Spalten i und j des Bildes. Intuitiv beschrieben, wird in der Formel der Kernel zeilenweise über den Input geschoben und ein repräsentativer Wert an der Stelle $[i,j]$ über das Matrixprodukt aus Kernel und Ausschnitt des Inputs berechnet.

In Abbildung 3–8 hat der Kernel der Convolutions die Dimensionen 5×5. Das führt dazu, dass die Feature-Map nach der Convolution kleiner ist als das Ausgangsbild bzw. die Feature-Map aus dem vorigen Layer, da sich der Kernel beim Schieben über das Bild nicht über den Bildrand hinaus bewegen kann.

Das Prinzip der Convolution wird in Abbildung 3–9 visualisiert. In dieser Abbildung wird sich einer Technik namens *Zero-Padding* bedient, welche das Bild bzw. das Array $\boldsymbol{I}$ am Rand mit Nullen auffüllt, damit die Interaktion des Kernels mit Pixeln im Randbereich definiert ist. Dadurch wird erzielt, dass der originale Input $\boldsymbol{I}$ und das Ergebnis der Faltung $\boldsymbol{S}$ gleich groß sind, was in Abbildung 3–8, wie oben besschrieben, nicht der Fall ist.

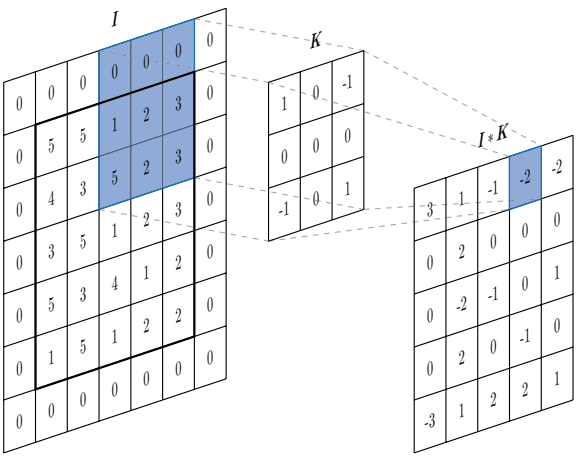


Abbildung 3–9: Convolution an einem beispielhaften zweidimensionalen Array mit Zero-Padding (modifiziert aus Frochte (2020))

Wichtig ist anzumerken, dass Gleichung (3–3) analog zu Abbildung 3–8 für nur einen Farbkanal gilt. Für Farbbilder mit drei Farbkanälen erhalten die Matrizen $\boldsymbol{S}$, $\boldsymbol{I}$ und $\boldsymbol{K}$ entsprechend eine weitere Dimension. Ebenso werden typischerweise mehrere Kernel in einem Layer herangezogen, um verschiedene Features detektieren zu können. Nach der Anwendung des Kernels wird häufig zusätzlich, analog zu den MLPs, eine Aktivierungsfunktion auf das Ergebnis angewandt.

Subsampling: Ein weiterer wichtiger Layer in einem CNN ist der *Subsampling*-Layer. Dessen Aufgabe ist es, die räumlichen Dimensionen der Feature-Maps zu reduzieren und dabei das wichtigste Feature aus einer Gruppe einzelner Features zu extrahieren. Dadurch wird die Unabhängigkeit des Features von der exakten räumlichen Positionierung gefördert. In der Praxis wird häufig das sog. *Max-Pooling* eingesetzt, welches eine Feature-Map in kleine $(z \times z)$-Domänen unterteilt und der Maximalwert innerhalb dieser Domäne als repräsentativer Wert für die gesamte Domäne eingesetzt wird. Das Max-Pooling wird in Abbildung 3–10 exemplarisch dargestellt. Sowohl für die Convolution, als auch für das Pooling, kann die Schrittgröße (englisch: *Stride*) angepasst werden, mit der der Convolution-Kernel bzw. der Pooling-Kernel über den Input geschoben wird. Das beeinflusst die Größe der resultierenden Feature-Maps.

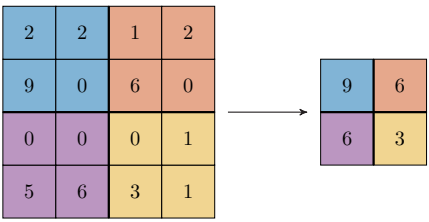


Abbildung 3–10: Max-Pooling (2×2) eines zweidimensionalen Arrays (modifiziert aus Frochte (2020))

Head: Nach einigen Convolutions und Poolings wird das CNN durch den *Head* vollendet. Dieser beinhaltet ein *Flatten*-Layer, auf den ein vollständiges MLP folgt. Innerhalb des Flatten-Layers werden die Einträge in den mehrdimensionalen Feature-Maps zu einem Vektor konkateniert, welcher direkt als Input in das MLP übergeben werden kann. Während die Feature-Maps als zentrale Aufgabe die Extraktion von Features aus dem Bild haben, erlaubt das MLP eine Verarbeitung der Features zur finalen Vorhersage im Downstream-Task.

3.4 Reinforcement Learning

In den folgenden Abschnitten wird das *Reinforcement Learning* (RL) grundlegend beschrieben. Diese Abschnitte sind angelehnt an die Ausarbeitung von Sutton und Barto (2020). Dabei ist der Inhalt auf die zum Verständnis dieser Arbeit relevanten Konzepte gekürzt und die mathematische Formulierung leicht angepasst.

3.4.1 Konzeptionelle Einführung in das Reinforcement Learning

RL beschreibt eine Methodik, bei der ein Agent selbstständig eine Strategie (englisch: *policy*) erlernt, um mit einer ihm zunächst unbekannten, möglicherweise nicht-stationären und/oder stochastischen Umgebung (englisch: *environment*) zu interagieren und so eine festgelegte Aufgabe zu erfüllen. Ein Agent ist in diesem Kontext ein Modell, welches Informationen entgegennimmt, verarbeitet und darauf basierend eine Ausgabe generiert (Ertel 2016). Dieser ist für die Entscheidungsfindung entsprechend seiner Strategie zuständig.

Die Strategie erlernt der Agent dadurch, dass die vom Agenten gewählten Aktionen von der Umgebung bewertet werden. Dazu verteilt die Umgebung numerische Rewards bzw. Belohnungen nach jeder durchgeführten Aktion, welche den Agenten in seinem Verhalten in Abhängigkeit der Aufgabe bestärken oder bestrafen. Aktionen ermöglichen es dem Agenten, Einfluss auf die Umgebung zu nehmen. Repräsentiert werden Aktionen i. d. R. von Vektoren mit skalaren Einträgen. Die Aktionen werden von dem Agenten basierend auf einer *Observation* bzw. Beobachtung des Zustands der Umgebung ausgewählt. Dabei sind Observationen typischerweise binäre, skalare, vektorielle oder tabellarische Repräsentationen des eigentlichen zugrundeliegenden Zustands.

Eine Besonderheit des RLs ist, dass nicht nur ausschließlich der Reward der aktuellen Aktion für den Agenten von Interesse ist, sondern auch die Rewards, die durch zukünftige Aktionen und den resultierenden Folgezuständen entstehen. Umgesetzt wird diese Weitsichtigkeit dadurch, dass der Agent nicht nur den aktuellen Reward, sondern einen kumulativen Reward über verschiedene Aktionen hinweg maximieren soll. Das birgt aber auch die Schwierigkeit der sog. *verzögerten Rewards*: Es ist möglich, dass ein Agent zunächst scheinbar suboptimal handeln muss, um die Gesamtperformance langfristig durch einen großen, erst später eingesammelten Reward zu maximieren. Das erschwert das Training des Agenten deutlich, da das Extrahieren der Aktionen, die zu den großen Rewards geführt haben, komplex ist.

Der *RL-Zyklus* ergibt sich aus der Abfolge von der Wahl einer Aktion basierend auf der Observation des aktuellen Zustands der Umgebung, der Anwendung der Aktion in der Umgebung und der daraus folgenden neuen Observation der Umgebung. Dieser Zyklus wird in Abbildung 3–11 visualisiert.

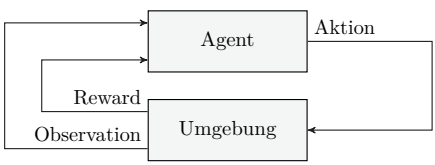


Abbildung 3–11: Visualisierung des RL-Zyklus (modifiziert aus Sutton und Barto (2020))

Das Training eines Agenten basiert auf einem Trial-and-Error-Ansatz, wodurch RL grundsätzlich eine hohe Sample-Komplexität besitzt. Zu Beginn des Trainings kennt der Agent die Umgebung und die Einflüsse der gewählten Aktionen auf die Umgebung nicht. Durch Ausprobieren verschiedener Aktionen erhält der Agent Feedback über die generierten Rewards und kann seine Strategie entsprechend anpassen. Alleiniges, zufälliges Ausprobieren von Aktionen ist allerdings nicht zielführend, um einen Agenten effizient zu trainieren. So würde der Agent das gelernte Wissen im Training nicht effizient ausnutzen. Dieser kann die Strategie über das bereits im lokalen Umfeld gesammelte Wissen hinaus nicht weiterentwickeln. Dieses Phänomen wird durch das *Exploration-Exploitation-Dilemma* beschrieben. *Exploration* beschreibt das Erkunden von neuen und ungesehenen Zuständen, welche den Zugang zu einer neuen und potentiell besseren Strategie eröffnen. *Exploitation* beschreibt das Ausnutzen von gelerntem Wissen. Der Agent verfolgt die gelernte Strategie, um das Training entsprechend dem aktuellen Wissensstand zu belohnungsreichen Zuständen zu bringen. Dadurch kann das bereits gelernte Wissen weiter verbessert werden.

Typische Einsatzgebiete des RLs sind das Anlernen von Robotern (Haarnoja et al. 2019), um beispielsweise selbstständig Laufen zu lernen, das Absolvieren von (Computer-)Spielen (Mnih et al. 2013) oder der Finanzsektor (Li et al. 2020; Wang et al. 2023), in dem beispielsweise Strategien zum Aktienhandel entwickelt werden. Grundsätzlich eignen sich alle Aufgabenstellungen, bei denen Entscheidungen basierend auf Zuständen automatisiert getroffen werden müssen. Die Optimierung von Crashstrukturen mittels RL ist ein neues Konzept, welches mit dieser Dissertation beleuchtet werden soll. Aufgrund der hohen Sample-Komplexität birgt der Einsatz von RL in der Crashmechanik mit hohen Simulationszeiten neue Herausforderungen, welche in dieser Dissertation untersucht werden.

3.4.2 Theoretische Grundlagen

Das theoretische Fundament des RLs stellen *Markow-Entscheidungsprobleme* (MEPs) (englisch: *markov decision processes*) dar. Bei einem MEP handelt es sich um einen Formalismus zur Beschreibung von Entscheidungsproblemen in dynamischen Umgebungen, deren Zustandsübergänge probabilistisch sind. Dabei sind den Zustandsübergängen Rewards zugeordnet. Eine Zustandsänderung wird von einem Agenten durch Wahl einer Aktion bewirkt. Ziel ist es, für gegebene Zustände der Umgebung, den optimalen Entscheidungsprozess zu finden, der dazu führt, dass der Agent im Durchschnitt den höchstmöglichen kumulativen Reward erhält.

Ein MEP wird über das Tupel $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ definiert. Der Agent interagiert zu diskreten Zeitpunkten $t \in \mathbb{N}_0$ mit der durch das MEP definierten Umgebung. Dabei ist

- $\mathcal{S}$ der Zustandsraum, also die Menge an möglichen Zuständen, die der Agent beobachten kann,

- $\mathcal{A}$ der Aktionsraum, d. h. die Menge an möglichen Aktionen, die der Agent ausführen kann,
- $\mathcal{P}$ die Menge an Übergangswahrscheinlichkeiten, welche angeben, mit welcher Wahrscheinlichkeit man von Zustand s durch Aktion a in den nächsten Zustand s' gelangt und
- $\mathcal{R}$ die Menge an reellwertigen Rewards, die einem Zustandswechsel von s nach s' direkt zugewiesen werden.

Da es sich bei einem MEP um einen probabilistischen Prozess handelt, werden die genannten Größen als Zufallsvariablen aufgefasst. Die zugehörigen Zufallsvariablen lauten S_t zur Beschreibung der Zustände, A_t zur Beschreibung der Aktionen und R_t zur Beschreibung des Rewards zum Zeitpunkt t. Konkrete Instanzen, die die Zufallsvariablen annehmen können, heißen $s \in \mathcal{S}$, $a \in \mathcal{A}$ und $r \in \mathcal{R}$. Analog zur Literatur wird der Zustand, der aus Zustand s mit Aktion a erreicht wird, s' genannt. Damit lässt sich die Menge an Übergangswahrscheinlichkeiten formell als $\mathcal{P} := \Pr\left(S_{t+1} = s' \mid S_t = s, A_t = a\right)$ beschreiben. Eine deterministische Strategie $\pi(s)$ ordnet einem Zustand s eine Aktion a zu. Eine probabilistische Strategie $\pi\left(a \mid s\right)$ gibt die Wahrscheinlichkeit an, Aktion a im Zustand s auszuwählen.

Im Rahmen eines MEPs wird in der Literatur meist von Zuständen gesprochen. Dabei beinhaltet der Zustand eine vollständige Beschreibung der aktuellen Konfiguration der Umgebung. In realen Problemstellungen ist der Zustand meist nicht bekannt. Stattdessen wird der Zustand durch eine Beschreibung des Zustands, der Observation, repräsentiert. Dafür eignen sich insbesondere die später besprochenen Approximationsverfahren. Bei einem MEP wird davon ausgegangen, dass die Wahrscheinlichkeit zum Erreichen von s' nur durch s selbst und nicht von vorigen Zuständen abhängig ist. Diese Anforderung an die Zustände nennt sich Markow-Annahme. Relevante Informationen aus vorigen Zuständen müssen im aktuellen Zustand inkludiert sein.

Bisher wurde das Framework vorgestellt, welches die Umgebung definiert, mit der der Agent interagiert. Zentraler Bestandteil für das Training eines Agenten ist der Return bzw. Gewinn

$$G_t := R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T, \tag{3–4}$$

den dieser durch Interaktion mit der Umgebung ausgehend nach Zeitpunkt t erzielt. Dabei handelt es sich um eine Summe über die vom Agenten in dem Zustand s_t zukünftig gesammelten Rewards bis zu einem terminalen Zeitpunkt T, welcher durch die Umgebung vorgegeben wird. Für $T \in \mathbb{N}$ spricht man von episodischen Problemen, wobei einzelne Episoden unabhängig voneinander sind. Ein typisches Beispiel hierfür sind (Video-)Spiele, bei denen zum Zeitpunkt T das Spielende erreicht ist und eine neue Spielrunde gestartet wird. Eine Spielrunde entspricht hierbei einer Episode.

Im Fall von kontinuierlichen Problemen gilt $T = \infty$. Der Agent interagiert theoretisch unendlich lang mit der Umgebung, ohne, dass ein terminaler Zustand erreicht wird. Ein Beispiel für eine kontinuierliche Problemstellung ist ein Roboter, welcher im Dauerbetrieb eingesetzt werden soll. Damit die Reihe (3–4) nicht divergiert, wird für den Fall einer kontinuierlichen Problemstellung eine diskontierte Variante des Returns vorgestellt. Der diskontierte Return wird definiert als

$$\begin{aligned} G_t &:= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad \text{für } 0 < \gamma \leq 1 \\ &= R_{t+1} + \gamma G_{t+1}. \end{aligned} \tag{3–5}$$

Dabei ist γ der Diskontierungsfaktor und beschreibt, wie stark zukünftige Rewards in die Entscheidungsfindung im aktuellen Zustand eingehen sollen. Ein Diskontierungsfaktor von $\gamma = 1$ bedeutet für den Return G_t, dass *alle* dem Zeitpunkt t folgenden Rewards gleiche Wichtigkeit besitzen. Dieser Fall ist analog zu der Reihe (3–4). Für einen Diskontierungsfaktor $0 < \gamma < 1$ werden zukünftige Rewards schwächer gewichtet und der Agent erachtet somit zukünftige Zustände als weniger relevant. Je kleiner der Diskontierungsfaktor γ gewählt wird, desto weniger Relevanz haben zukünftige Zustände. Für kontinuierliche Probleme wird ein Diskontierungsfaktor $\gamma < 1$ vorausgesetzt, um die Beschränktheit der Reihe (3–5) zu gewährleisten. Ein kleinerer Diskontierungsfaktor erhöht die Trainingsstabilität, da der Einfluss von weit entfernten, möglicherweise unsicheren Rewards reduziert wird. Gleichzeitig kann ein kleiner Diskontierungsfaktor dazu führen, dass der Agent suboptimale langfristige Strategien lernt, weil er nicht genug Wert auf zukünftige Rewards legt. Es ist auch möglich, dass der Diskontierungsfaktor $\gamma = 0$ gewählt wird. Dann gilt für den Return $G_t = R_{t+1}$. Dann ist ausschließlich der direkt folgende Reward für die Entscheidungsfindung des Agenten verantwortlich. In diesem Fall nennt man dann das Verhalten des Agenten *myopisch* bzw. kurzsichtig. Zwar folgt die Reihe (3–5) argumentativ daraus, dass die Konvergenz der Reihe in kontinuierlichen Problemstellungen gewährleistet bleiben soll, aber in der Praxis wird diese Reihe häufig auch in episodischen Problemstellungen eingesetzt, um mehr Kontrolle über die anzulernende Strategie zu haben.

Eine Eigenschaft der Reihe (3–5), welche insbesondere für die Entwicklung von RL-Algorithmen wichtig ist, ist deren Rekursivität. Diese erlaubt das Entwickeln von effizienten Algorithmen, welche nicht darauf angewiesen sind die unendliche Reihe stets neu zu berechnen.

Um das Training eines Agenten besser formulieren zu können, ist es sinnvoll zu quantifizieren, wie wertvoll verschiedene Zustände der Umgebung für den Agenten sind. Dafür wird eine Zustandswertfunktion $v_\pi(s)$ (englisch: *state-value function*) eingeführt. Sie quantifiziert den erwarteten Return, wenn ein Agent seiner Strategie π folgt und in

Zustand s startet. Die Zustandswertfunktion lautet

$$\begin{aligned} v_\pi(s) &:= \mathbb{E}_\pi \left[G_t \,|\, S_t = s \right] \\ &= \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k R_{t+k+1} \,\middle|\, S_t = s \right]. \end{aligned} \tag{3–6}$$

Dabei ist $\mathbb{E}_\pi [\cdot]$ der Erwartungswert einer Zufallsvariable. Das Subskript π gibt an, dass der Agent strikt nach der Strategie π handelt.

Analog wird eine Funktion definiert, die den Wert einer Aktion a des Agenten bestimmt. Diese Funktion wird Aktionswertfunktion $q_\pi(s, a)$ (englisch: *action-value function*) genannt und lautet

$$\begin{aligned} q_\pi(s, a) &:= \mathbb{E}_\pi \left[G_t \,|\, S_t = s, A_t = a \right] \\ &= \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a \right]. \end{aligned} \tag{3–7}$$

Der Unterschied zur Zustandswertfunktion besteht darin, dass ausgehend vom Zustand s nicht direkt entsprechend der Strategie vorgegangen wird, sondern erst Aktion a ausgeführt wird und dann nach der Strategie verfahren wird.

Eine optimale Strategie π_* maximiert die Zustandswertfunktion

$$v_*(s) := \max_\pi v_\pi(s) \quad \forall s \in \mathcal{S} \tag{3–8}$$

bzw. maximiert die Aktionswertfunktion

$$q_*(s, a) := \max_\pi q_\pi(s, a) \quad \forall s \in \mathcal{S} \text{ und } \forall a \in \mathcal{A}. \tag{3–9}$$

3.4.3 Grundlegende Lernverfahren

Bisher wurde definiert, was der Agent lernen soll. Wie eine Lösung entsprechend der Definition eines MEPs gefunden werden kann, wird in diesem Abschnitt beschrieben. Methoden der *dynamischen Programmierung* stellen die Basis für viele RL-Algorithmen dar. Dabei wird das Optimierungsproblem in Teilprobleme aufgeteilt, um es einfacher lösen zu können. Solche Methoden setzen ein explizites Modell eines idealen MEPs voraus, was bedeutet, dass die Übergangswahrscheinlichkeiten und Rewards der Umgebung für den Algorithmus bekannt sein müssen. Ebenso wird vorausgesetzt, dass der Zustands- und Aktionsraum endlich ist. Falls das nicht der Fall sein sollte, lassen sich Näherungslösungen durch Quantisieren der Problemstellung finden. Zu den Methoden der dynamischen Programmierung gehört das Policy-Iteration- und das Value-Iteration-Verfahren. Die in Abschnitt 3.4.2 vorgestellten Wertfunktionen $v_\pi(s)$ und $q_\pi(s, a)$ stellen die Grundlage für die Suche nach einer optimalen Strategie dar.

Da das Policy-Iteration-Verfahren zentraler Bestandteil der im Folgenden beschriebenen Methoden ist, wird dieses näher erläutert. Es wechseln sich das Evaluieren der aktuellen Strategie (englisch: *policy evaluation*) und das Verbessern der aktuellen Strategie (englisch: *policy improvement*) ab. Durch das Zusammenspiel konvergiert die Strategie gegen die optimale Strategie. Unter Ausnutzung des rekursiven Zusammenhangs $G_t = R_{t+1} + \gamma G_{t+1}$ lässt sich die Zustandswertfunktion $v_\pi(s)$ in Form der sog. Bellman-Gleichung formulieren als

$$\begin{aligned} v_\pi(s) &:= \mathbb{E}_\pi \left[G_t \,|\, S_t = s \right] \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \,|\, S_t = s \right] \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \,|\, S_t = s \right]. \end{aligned} \tag{3–10}$$

Aus dieser Bellman-Gleichung lässt sich eine Iterationsvorschrift zur Bestimmung der Zustandswertfunktion $v_\pi(s)$ ableiten. Die Zustandswertfunktion wird bei dem Policy-Iterations-Verfahren genutzt, um die aktuelle Performance der Strategie π zu evaluieren. Zum Verbessern der aktuellen Strategie eignet sich die Betrachtung der Bellman-Gleichung der Aktionswertfunktion $q_\pi(s, a)$ auf Basis der Zustandswertfunktion $v_\pi(s)$

$$q_\pi(s, a) := \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \,|\, S_t = s, A_t = a \right]. \tag{3–11}$$

Mithilfe der Bellman-Gleichung der Aktionswertfunktion (3–11) kann eine neue Strategie π' bestimmt werden, die stets besser oder mindestens genauso gut wie die aktuelle Strategie π ist. Dazu wird die Aktion ausgewählt, die die Aktionswertfunktion für einen Zustand maximiert. Das lässt sich formulieren als

$$\pi'(s) := \underset{a}{\operatorname{argmax}}\, q_\pi(s, a) \tag{3–12}$$

und nennt sich gierige (englisch: *greedy*) Strategie. Die strikte Verbesserung der Strategie ist sichergestellt durch die Konstruktion der Gleichung (3–11), da nach der Aktion a der Zustandswert v_π der alten Strategie π herangezogen wird.

Ein großer Nachteil der Methoden der dynamischen Programmierung ist, dass das Modell explizit bekannt sein muss, d. h. die Rewards und Übergangswahrscheinlichkeiten der Umgebung müssen bekannt sein. Für die meisten praktischen Aufgabenstellungen ist das nicht gegeben. *Monte-Carlo-Methoden* kommen auch ohne diese Annahmen aus, da sie die Wertfunktionen $v_\pi(s)$ und $q_\pi(s, a)$ durch gesammelte Erfahrung (englisch: *experience*), das sind episodische Samples, abschätzen. Naturgemäß werden im Folgenden episodische Aufgabenstellungen betrachtet.

Das Abschätzen der Zustandswertfunktion $v_\pi(s)$ zur Evaluation geschieht durch eine Mittelwertbildung über die bisher gesammelten Returns, die aus dem aktuellen Zustand hervorgehen – der Erfahrung. Je mehr Returns gesammelt werden, desto besser approximiert der Mittelwert den wahren Zustandswert. Unter Betrachtung der Zustandswertfunktion $v_\pi(s)$ aus Gleichung (3–6) lässt sich erkennen, dass sich der Erwartungswert zur Bestimmung von v_π ohne ein explizites Modell nicht bestimmen lässt. Stattdessen

wird der Erwartungswert der Returns durch die oben beschriebene Mittelwertbildung approximiert. Eine Beispielvorschrift zur Bildung eines angepassten Mittelwertes lautet

$$V(S_t) \leftarrow V(S_t) + \alpha \left[G_t - V(S_t) \right], \tag{3–13}$$

wobei das groß geschriebene V signalisiert, dass es sich um einen approximierten Wert von v_π handelt. α ist ein Schrittweiten-Parameter, welcher steuert, wie stark $V(S_t)$ in einer Iteration angepasst wird. Wichtig ist, dass der Iterationsschritt erst durchgeführt werden kann, wenn die Episode durchlaufen ist und G_t bestimmt werden kann.

Returns können sich ausgehend von einem Zustand durch das Trainieren der Strategie ändern. Man spricht von einer nicht-stationären Verteilung der Returns. Die Glättung des Mittelwerts bewirkt, dass neu gesammelte Returns stärker gewichtet werden als Returns, die zu Beginn des Trainings gesammelt werden. Damit kann das Problem der nicht-stationären Verteilung der Returns gelöst werden.

Wie bereits diskutiert wurde, wird mit der Zustandswertfunktion $v_\pi(s)$ beschrieben, wie wertvoll ein Zustand s für den Agenten ist. Zur Bewertung der Performance einer Strategie ist das zwar ausreichend, aber ohne explizites Modell der Umgebung, alleinig basierend auf episodischen Samples, ist die Zustandswertfunktion $v_\pi(s)$ ungeeignet, um die aktuelle Strategie zu verbessern. Sinnvoll hingegen ist es, die Aktionswertfunktion $q_\pi(s, a)$ für Zustands-Aktions-Paare zu approximieren, da diese zusätzlich die Information enthält, welche Aktion in einem Zustand den höchsten Wert hat. Der Prozess der Optimierung der Strategie funktioniert ähnlich zu dem Policy-Iteration-Verfahren der dynamischen Programmierung.

Die Betrachtung der approximierten Aktionswertfunktion $Q_\pi(s, a)$ birgt den Nachteil, dass bei einer deterministischen Strategie für gleiche Zustände stets die gleiche Aktion gewählt wird und der Agent aufgrund fehlender Exploration nicht alle Zustands-Aktionswert-Paare sammeln kann. Somit kann der Agent nur eine unzureichende Approximation von $q_\pi(s, a)$ bilden. Hier wird in der Praxis häufig eine ϵ-greedy-Strategie benutzt, die das Erkunden der Umgebung und das Ausnutzen des gelernten Wissens in ein Gleichgewicht bringen soll. Dazu wird bei der Wahl einer Aktion mit einer von ϵ abhängenden Wahrscheinlichkeit eine Zufallsaktion gewählt und nicht nach der Strategie des Agenten gehandelt, um den Agenten die Umgebung explorieren zu lassen. Alternativ werden häufig auch direkt stochastische Strategien, auch weiche Strategien genannt, verwendet. Dabei werden die Aktionen entsprechend einer probabilistischen Verteilung $\pi(a \mid s)$ ausgewählt.

Eine Kombination der dynamischen Programmierung und der Monte-Carlo-Methoden sind *Temporal Difference*-Methoden (TD-Methoden) bzw. das TD-Lernen. Bei Monte-Carlo-Methoden muss die Methode auf das Beenden einer Episode warten, um den Return der Episode zu ermitteln und somit zu evaluieren, wie gut die Strategie ist. Dadurch konvergieren Agenten mit dieser Methode nur langsam. TD-Methoden hingegen lernen bereits durch einzelne Schritte in der Umgebung, sodass die Methoden auch auf

kontinuierliche Problemstellungen angewandt werden können. Dieses Vorgehen, bei dem aktuelle Zustandswerte basierend auf geschätzten, zukünftigen Zustandswerten ermittelt werden, wird als *Bootstrapping* bezeichnet. Analog zur Iterationsvorschrift (3–13) bei den Monte-Carlo-Methoden lautet eine einfache Vorschrift für die Evaluierung einer mit TD-Methoden angelernten, zustandswertbasierten Strategie

$$\begin{aligned} V(S_t) &\leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right] \\ &= V(S_t) + \alpha \delta_t \end{aligned} \tag{3–14}$$

Aus der Vorschrift geht hervor, dass nur der Zustandswert für den nächsten Zustand benötigt wird und nicht der Return der gesamten Episode wie bei den Monte-Carlo-Methoden. Dabei wird $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ für zustandswertbasierte Strategien häufig auch als TD-Fehler δ_t bezeichnet und spielt in vielen modernen RL-Algorithmen eine zentrale Rolle. Der TD-Fehler ist die Differenz zwischen der aktuellen Schätzung des Zustandswerts und einer neuen Schätzung, die auf dem nächsten Zustand und seinem Zustandswert basiert.

Zuletzt wird eine wichtige Differenzierung zweier Arten an Algorithmen des RLs vorgestellt:

- *On-Policy-Algorithmen* lernen direkt von den aktuellen Interaktionen des Agenten mit der Umgebung basierend auf der aktuellen Strategie des Agenten. Dies bedeutet, dass der Agent die Konsequenzen seiner aktuellen Strategie direkt erfährt und von diesen Interaktionen lernt. Ein Nachteil von reinen On-Policy-Methoden ist, dass sie häufig weniger effizient sind, da sie nur von den neuesten Samples lernen und ältere Erfahrungen nicht wiederverwenden. Konträr dazu zeichnen sie sich aber häufig auch durch eine höhere Stabilität beim Training aus, da alle Informationen aus der aktuellen Strategie folgen.
- *Off-Policy-Algorithmen* sind so konzipiert, dass sie von Erfahrung lernen können, die innerhalb einer anderen Strategie als der aktuellen gesammelt wurden. Dies ermöglicht es, sowohl aktuelle als auch alte Erfahrungen zu nutzen und möglicherweise schneller zu lernen oder mehr von den verfügbaren Samples zu profitieren.

3.4.4 Approximationsverfahren

Bei den bisher vorgestellten Methoden zur Bestimmung einer optimalen Strategie handelt es sich um tabellarische Verfahren. So enthält die Tabelle beispielsweise für alle Zustands-Aktions-Paare den zugehörigen Aktionswert. Für praktische Aufgabenstellungen ist das aus folgenden Gründen problematisch (Sutton und Barto 2020):

- Bei großen Zustands- und Aktionsräumen, wie es bei praxisnahen Aufgabenstellungen typisch ist, reicht der Speicherplatz für die Tabelle der Zustands- bzw. Aktionswerte praktisch nicht aus. Das lässt sich leicht erkennen, wenn beispielsweise

Bilder eines Videospiels den Zustand einer Umgebung beschreiben. Nicht nur die Anzahl der Zustände und somit der Bilder ist in einer solchen Umgebung enorm. Für jeden einzelnen Zustand muss dann für die Aktionswerte auch noch der Wert für die Kombination des Zustands mit jeder möglichen Aktion abgespeichert werden.

- Die große Menge an Zuständen und Aktionen bewirkt, dass die einzelnen Werte für Zustands-Aktions-Paare in einer praktikablen Zeit nicht genau genug angelernt werden können. Abhängig von der Umgebung kann es passieren, dass viele Zustände nur einmal besucht werden und keine sinnvolle Approximation für die Zustands- und Aktionswerte gefunden werden kann. Somit kann keine optimale Strategie angelernt werden.

Eine Möglichkeit diese Probleme zu umgehen sind *Funktionsapproximatoren*. Dabei handelt es sich um parametrisierte Funktionen, die funktionale Zusammenhänge zwischen Eingangs- und Ausgangsgrößen abbilden können. Im Falle der Evaluierung einer Strategie kann beispielsweise eine Funktion $\hat{v}(s, \boldsymbol{w}) \approx v_\pi(s)$ gefunden werden, die einer Näherung der Zustandswertfunktion für Zustand s unter dem Parametervektor $\boldsymbol{w} \in \mathbb{R}^d$ entspricht. Durch den Zusammenhang zwischen Zustand und approximierter Zustandsfunktion müssen Zustandswerte nicht mehr tabellarisch abgespeichert werden, sondern können direkt aus dem funktionalen Zusammenhang bestimmt werden. Dadurch, dass die Komplexität der approximierenden Funktion $\hat{v}(s, \boldsymbol{w})$ durch die Anzahl ihrer Parameter begrenzt ist, generalisiert die Funktion das Verhalten der Zustandswertfunktion $v_\pi(s)$. Die Änderung eines Parameters im Parametervektor bewirkt damit eine Änderung für viele verschiedene Zustände. Die Approximation der Zustandswertfunktion erlaubt es ebenso, dass RL-Algorithmen auch mit partiellen Observationen zurechtkommen. Diese beinhalten, anders als im vorigen Abschnitt stets gefordert ist, nicht alle Informationen, um den Zustand vollständig zu beschreiben.

Damit die Approximation die originale Funktion möglichst gut abbilden kann, muss ein entsprechender Parametervektor $\boldsymbol{w}$ gefunden werden. Es gibt verschiedene Ansätze eine approximierende Funktion zu bilden. Dabei wird zwischen linearen und nichtlinearen Funktionen unterschieden. Die für diese Arbeit relevantesten Funktionsapproximatoren sind NNs, so wie sie in den Abschnitten 3.3.1 und 3.3.2 vorgestellt wurden. Sie zeichnen sich durch ihre große Flexibilität und Universalität aus, was hilfreich für komplexe Problemstellungen ist.

Alle bisher gezeigten Algorithmen des RL basierten darauf, dass die Aktionswertfunktion angelernt wird und der Agent dann eine Strategie entsprechend der Aktionswertfunktion verfolgt. In einem anderen, neuen Ansatz wird eine parametrisierte Strategie direkt angelernt, wobei grundlegende Konzepte aus den bisher ausgeführten Abschnitten auch für diesen Ansatz genutzt werden. Diese Strategie wählt Aktionen aus, ohne dabei direkt auf Wertfunktionen zurückzugreifen. Das Verfahren nennt sich *Policy-Gradient-Methode*. Die Parameter $\boldsymbol{\theta}$ der Strategie müssen entsprechend einer Metrik $J(\boldsymbol{\theta})$ optimiert werden. Dabei ist die Parametrisierung wie bei den wertebasierten Funktionsapproximatoren

flexibel. Optimiert wird der Parametervektor mit dem Gradientenanstiegsverfahren mit der Iterationsvorschrift

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla J(\boldsymbol{\theta}_t). \tag{3–15}$$

Dabei ist α ein Schrittweiten-Parameter, welcher im Kontext des MLs zur Skalierung der Gradienten häufig auch als *Lernrate* bezeichnet wird. Die Gradienten im Punkt $\boldsymbol{\theta}_t$ werden durch $J(\boldsymbol{\theta}_t)$ angegeben. Ein Beispiel für eine Metrik $J(\boldsymbol{\theta})$ in einer episodischen Problemstellung ist

$$J(\boldsymbol{\theta}) \coloneqq v_{\pi_{\boldsymbol{\theta}}}(s_0) \tag{3–16}$$

ausgehend von einem Startzustand s_0 der Episode. Bei dieser Metrik wird auf die Zustandswertfunktion $v_{\pi_{\boldsymbol{\theta}}}(s_0)$ zurückgegriffen. Diese wird aber nur zum Optimieren der Parameter der Strategie benutzt und wird nicht, wie bisher, bei einer konkreten Entscheidungsfindung des Agenten herangezogen.

Vorteile dieser Herangehensweise gegenüber den wertbasierten Verfahren sind die Robustheit gegenüber Approximationsfehler, da die Strategie direkt optimiert wird und dadurch weniger sensitiv bezüglich Fehlern in der Wertfunktion ist. Zudem besteht die Möglichkeit zur flexiblen Anpassung der Performance-Metrik $J(\boldsymbol{\theta})$, die auch die Integration von domänen-spezifischem Wissen erlaubt. Zudem eignen sich direkt optimierte Strategien auch für kontinuierliche Aktionsräume. Große Nachteile hingegen sind eine langsame Konvergenz der Verfahren und eine höhere Ineffizienz bezüglich der benötigten Samples.

Actor-Critic-Algorithmen (Konda und Tsitsiklis 1999) stellen eine Kombination der wertbasierten und der strategiebasierten Methoden dar und sind für diese Dissertation besonders relevant. Als Form des TD-Lernens sind sie rechnerisch effizient, können mit großen Zustands- und Aktionsräumen umgehen und sind robust gegenüber verrauschten oder unvollständigen Observationen. Darüber hinaus kann es sowohl für das Lernen diskreter als auch kontinuierlicher Strategien verwendet werden.

Die Actor-Critic-Algorithmen bestehen aus zwei zentralen Komponenten: einem Actor, der lernt, sinnvolle Aktionen in einem Zustand auszuwählen und einem Critic, der lernt, den Zustandswert des Zustands vorherzusagen. Sowohl der Actor als auch der Critic sind dabei Funktionsapproximatoren. Die Aufgabe des Critics ist es, den Actor in seiner Entscheidungsfindung zu kritisieren. In Abbildung 3–12 wird die Architektur eines Actor-Critic-Algorithmus als Bestandteil des RL-Zyklus (vgl. Abbildung 3–11) aufgezeigt.

Basierend auf einem beobachteten Zustand wählt der Actor eine Aktion aus und übergibt diese an die Umgebung. Die Umgebung antwortet darauf mit einem neuem Zustand und einem Reward. Der Zustand wird als Input an den Critic übergeben, welcher daraufhin eine Vorhersage über den Zustandswert trifft.

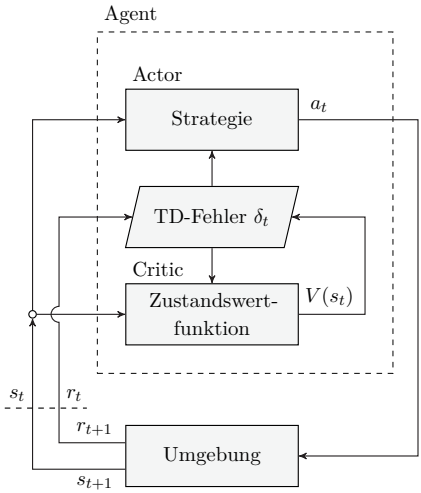


Abbildung 3–12: Architektur eines Actor-Critic-Algorithmus (modifiziert aus Sutton und Barto (2015))

Zusammen mit dem Reward kann der TD-Fehler $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ bestimmt werden. Bei positivem TD-Fehler hat die gewählte Aktion zu einem besseren Zustandswert geführt, als das, was der Critic geschätzt hat. Bei negativem TD-Fehler sollte die Aktion in vergleichbaren Zuständen zukünftig weniger genutzt werden, da der Critic einen höheren Zustandswert erwartet hat. Der TD-Fehler wird direkt genutzt, um den Critic zu verbessern. Praktisch wird dafür beispielsweise der mittlere quadratische TD-Fehler minimiert.

Zusätzlich wird der TD-Fehler an den Actor übergeben. Darüber erhält der Actor Informationen darüber, ob die gewählte Aktion gut oder schlecht war. Die konkrete Zielfunktion zur Optimierung des Actors hängt stark von dem gewählten Trainingsalgorithmus ab, wobei der TD-Fehler einen wichtigen Beitrag in der Zielfunktion bildet. Das Training des Actors erfolgt mittels der Policy-Gradienten-Methode.

4 Stand der Technik zur Entwicklung von Crashstrukturen gestützt durch Machine Learning

In diesem Kapitel wird ein Überblick bezüglich der Kombination von Methoden der KI- und der simulationsbasierten Entwicklung und Optimierung von Crashstrukturen in der Fahrzeugentwicklung aufgezeigt. Der Fokus liegt dabei vor allem auf der konkreten ingenieurtechnischen Anwendung, die ML-gestützte Methoden im Rahmen der Entwicklung von Crashstrukturen ermöglichen.

In Abschnitt 4.1 werden Methoden diskutiert, wie Rohdaten von Simulationen (z. B. netzabhängige FE-Daten) in ML-taugliche Formate transformiert werden können. Mit Abschnitt 4.2 folgen dann verschiedene Anwendungsbeispiele mit praktischem ingenieurtechnischem Bezug zur Anwendung der KI- bzw. ML-Methoden.

4.1 Aufbereitung von Simulationsdaten

Ergebnisse aus FE-Simulationen sind netzabhängig. An FE-Knoten oder FE-Elementen werden Strukturantworten, wie Verschiebungen und Dehnungen, ausgelesen. Die direkte knoten- und elementbasierte Vergleichbarkeit von Netzen ist i. Allg. nicht gegeben, insbesondere dann nicht, wenn geometrische Änderungen zwischen Modellvarianten vorliegen. Da ML-Modelle aber häufig auf strukturierten Daten arbeiten, können die unstrukturierten Simulationsergebnisse also nicht direkt als Feature für die ML-Modelle herangezogen werden.

Der einfachste Ansatz zur Umgehung des Problems der Netzabhängigkeit ist, dass das ML-Modell direkt die Geometrieparameter oder Entwurfsvariablen eines Modells als Features entgegen nimmt. Dadurch ist man unabhängig vom FE-Netz. Ein solcher Ansatz wird beispielsweise von Li et al. (2021) verfolgt. Diese Herangehensweise birgt aber große Nachteile: Der zugrundeliegende physikalische Zusammenhang zwischen den crashrelevanten Auswertegrößen aus der Simulation und der daraus resultierenden Vorhersagegröße des ML-Modells wird nicht direkt angelernt. Stattdessen wird der physikalische Zusammenhang implizit über die Entwurfsvariablen antrainiert. Dadurch

erhält man eine starke Abhängigkeit von der zugrundeliegenden Problemstellung. Da kein direkter Zusammenhang zu der eigentlichen Physik besteht, sind solche Modelle häufig sensitiver bzgl. der Modellperformance bei Änderung von Entwurfsvariablen über den angelernten Bereich hinaus. Außerdem kann die Aufgabenstellung für ein trainiertes Modell nicht ohne Weiteres angepasst werden. Kommt beispielsweise nachträglich eine neue Entwurfsvariable zu der Problemstellung hinzu, so muss auch das ML-Modell komplett neu trainiert werden.

Ein anderer, einfacher Ansatz wird von Hahner et al. (2020) mit *Oriented Bounding Boxes* (OBBs) vorgestellt. OBBs sind Quader, die das Bauteil zu jedem Zeitpunkt der FE-Simulation mit minimalem Volumen umschließen. Dadurch kann die Translation, Rotation und Größe von Crashstrukturen in Fahrzeugen zu verschiedenen Zeitpunkten abgeschätzt werden. Damit soll die plastische Deformation des Bauteils abstrakt abgebildet werden. Ob eine solch starke Abstraktion der Mechanik sinnvoll ist, muss individuell für das betrachtete Problem entschieden werden. Beispielsweise ist es schwierig über die OBBs Rückschlüsse auf lokales mechanisches Verhalten zu schließen.

Kracker et al. (2020) stellen zur Generierung eines strukturierten Datensatzes aus einem unstrukturierten FE-Netz zwei Ansätze vor. Zum einen können die FE-Elemente der betrachteten Struktur auf eine das Bauteil umschließende Kugelfläche projiziert werden. Diskretisiert werden die Daten auf der Kugelfläche über die beiden Polarkoordinaten, die die Position auf der Kugelfläche beschreiben. Dadurch wird das drei-dimensionale Problem in zwei Dimensionen beschrieben. Alternativ kann auch ein Quader genutzt werden, der das Bauteil umschließt und räumlich in Voxel diskretisiert ist. Hier werden die FE-Elemente dann auf die nächstgelegenen Voxel projiziert. Dadurch bleibt die Räumlichkeit der Daten auf Kosten der höheren Dimensionalität im Vergleich zum Ansatz mit der Kugelfläche erhalten.

Einen ähnlichen Ansatz basierend auf der Diskretisierung der Struktur in Voxel stellen Kohar et al. (2021) vor. Die Struktur ist von einem in Voxel diskretisierten Quader umschlossen. Wenn ein FE-Knoten in einem Voxel liegt, so wird dieser aktiviert. Die aktivierten Voxel halten einen Wert, der der normierten Masse innerhalb des Voxels entspricht. Somit repräsentieren die aktivierten Voxel die Struktur und deren Massenverteilung. Die Voxel werden mit einem 3D-CNN-Autoencoders in eine niedrigerdimensionale Einbettung überführt. Ein 3D-CNN funktioniert dabei analog zu dem in Abschnitt 3.3.2 gezeigten CNN mit einer dritten räumlichen Dimension. Autoencoder sind NNs, welche als Input Daten entgegennehmen und versuchen genau diese Daten wieder im Output vorherzusagen. Dabei ist die Anzahl der Neuronen im Hidden-Layer kleiner als im Input bzw. Output-Layer. Die Einbettung wird dann durch die Aktivierungen der Neuronen in dem kleinsten Hidden-Layer repräsentiert. Dadurch, dass den Datenpunkten keine Labels zugeordnet sind, werden Autoencoder dem UL zugeschrieben.

In Diez (2019) und Diez et al. (2016) werden primär eindimensional ausgeprägte Crashstrukturen, wie beispielsweise Crashboxen oder Längsträger, betrachtet. Zur geometrischen Reduktion des Modells wird eine Regressionslinie bestmöglich durch die Struktur gelegt, sodass sie den quadratischen Abstand zu den FE-Knoten minimiert. Die Regressionslinie wird als Bézier-Kurve parametrisiert. Dadurch können Strukturantworten, wie beispielsweise plastische Dehnungen, entlang der Struktur auf die Regressionslinie projiziert werden. Die projizierten Strukturantworten können dann auf der Regressionslinie beliebig diskretisiert werden.

Ein weiterer Ansatz funktioniert ähnlich zu einer Fourier-Transformation, die zeitliche Signale in Frequenzspektren zerlegt. Das Ergebnis der Fourier-Transformation sind Fourier-Koeffizienten, die angeben, welche Frequenz wie stark in einem Zeitsignal vorhanden ist. Von Iza-Teran und Garcke (2019) und Garcke und Iza-Teran (2017) wird vorgeschlagen, die Geometrie von Strukturen in einem analogen Verfahren durch Spektralkoeffizienten zu beschreiben. Man erhält als Ergebnis der Transformation so viele Spektralkoeffizienten, wie Knoten des FE-Modells vorhanden sind. Den größten Beitrag zur Rekonstruktion der originalen Geometrie generieren dabei die wenigen aber größten Koeffizienten. Je mehr Spektralkoeffizienten die Struktur beschreiben, desto besser wird die originale Geometrie repräsentiert. Durch die Wahl der größten Spektralkoeffizienten lässt sich die Geometrie folglich niederdimensional darstellen.

4.2 Ingenieurtechnische Problemstellungen

4.2.1 Vorhersage von Strukturantworten

Eine populäre Anwendung von KI ist das Vorhersagen von Strukturantworten, welche ansonsten durch aufwändige Simulationen ermittelt werden müssten. Solche Vorhersagen könnten dann beispielsweise in einer Optimierungsaufgabe eingesetzt werden, um die Optimierungsdauer um ein Vielfaches zu reduzieren, da ressourcenintensive Crashsimulationen durch schnelle ML-Modelle ersetzt werden können.

Von Kohar et al. (2021) wird das ML-Modell dazu genutzt, Strukturantworten von Crashboxen unter Einwirkung einer beweglichen starren Wand vorherzusagen. Es wurden hierzu 2058 Simulationen mit unterschiedlichen geometrischen Konfigurationen der Crashbox durchgeführt. Im Fokus der Auswertung ist der Kraft-Weg-Verlauf gemessen an der starren Wand und die Deformation des FE-Netzes. Grundlage für die Methode ist die in Abschnitt 4.1 vorgestellte Voxelstrategie zur Aufbereitung der FE-Daten. Die undeformierte, in Voxel diskretisierte Struktur wird an einen 3D-CNN-Autoencoder übergeben. Basierend auf einem LSTM können dann unter Angabe von wenigen zusätzlichen Informationen der Kraft-Weg-Verlauf der starren Wand und die Netzdeformation vorhergesagt werden.

Basierend auf zeitabhängigen Signalen von Sensoren im Fahrzeug, wie zum Beispiel Beschleunigungsmessungen, haben Koch et al. (2018) eine Methode entwickelt, die mithilfe von ML vorhersagt, welche Strukturkomponenten nach einem Low-Speed-Crash beschädigt sein werden. Das Vorgehen ist dabei sequentiell aufgebaut und beginnt damit, 806 Features je Zeitsignal zu extrahieren. Diese Features beinhalten beispielsweise die Länge des Signals, Minima und Maxima, aber auch Fourier-Koeffizienten des Signals. Für jede betrachtete Fahrzeugkomponente werden dann die relevantesten Features ermittelt. Auf Basis derer kann für jede Komponente dann ein RF-Modell trainiert werden, welches angibt, ob eine Komponente bei gegebenen Zeitsignalen beschädigt oder nicht beschädigt ist. Nach welchem Kriterium eine Komponente als beschädigt klassifiziert wird, lassen die Autoren offen.

In einer anderen Veröffentlichung haben Koch und Bäck (2018) diese zeitabhängigen Signale genutzt, um auf einen ungefähren Kollisionspunkt zwischen dem betroffenen Fahrzeug und einem Impaktor Rückschluss ziehen zu können. Dabei handelt es sich nicht um eine Vorhersage einer Strukturantwort im engeren Sinne. Da das prinzipielle Vorgehen dabei aber analog zu obiger Veröffentlichung von Koch et al. (2018) ist, wird es in diesem Kontext mit erwähnt. Als ML-Modell wird hier ein DT genutzt, welches aus den extrahierten Parametern der Zeitsignale den Auftreffpunkt vorhersagen kann.

4.2.2 Identifikation von Deformationsmodi

Die Identifikation von Deformationsmodi von Crashstrukturen ist ein weit verbreitetes Forschungsfeld. Mittels Methoden der KI lassen sich diese kategorischen Zustände automatisiert identifizieren. So können unerwünschte Deformationsmodi frühzeitig detektiert und vermieden werden. Ein zu vermeidender Deformationsmodus wäre beispielsweise eine axial belastete und global ausgeknickte Crashstruktur, die dadurch eine geringere Effizienz in der Energieabsorption vorweist als eine sich durch Faltenbeulen deformierende Struktur.

Deshalb filtern Li et al. (2021) unerwünschte Deformationszustände auf Basis verschiedener Klassifikationsmodelle, wie beispielsweise kNNs, RFs, SVMs und NNs, heraus. Dazu wird ein Datensatz aus Entwurfsvariablen als Inputs und Strukturantworten und Deformationsmodi als Outputs generiert. Auf diesem Datensatz werden die ML-Modelle trainiert. Das beste ML-Modell wird dann für eine Optimierung herangezogen. Die Optimierungsformulierung wird dabei so angepasst, dass der gewünschte Deformationsmodus als Restriktion aufgenommen wird und die optimierte Lösung gesichert dem erstrebenswerten Modus entspricht. Durch die Abhängigkeit des ML-Modells von den Entwurfsvariablen ist das Modell, wie bereits angesprochen wurde, empfindlich bezüglich Parameteränderungen.

Von Garcke und Iza-Teran (2015) wird zur Analyse eines gesamten Datensatzes aus vielen ähnlichen Crashsimulationen ein Ansatz vorgeschlagen, bei dem eine Ähnlichkeitsmatrix der Simulationen basierend auf deren resultierenden Verschiebungen an den FE-Knoten aufgebaut wird. Es wird ein Vektor für jede Simulation angelegt, in den für jeden Knoten die resultierende Verschiebung eingetragen wird. Die Ähnlichkeit von Simulationen für die Ähnlichkeitsmatrix kann dann über die euklidische Distanz der beiden Vektoren der zugehörigen Simulationen bestimmt werden. Da die Matrix in Abhängigkeit der Knotenanzahl des Modells sehr groß ist, wird anschließend eine Dimensionsreduktion angewandt, um Abhängigkeiten innerhalb der Matrix zu eliminieren und nur die wichtigsten Informationen beizubehalten. Die Einbettung kann dann beispielsweise visualisiert werden und erlaubt Einblick in das Deformationsverhalten von den betrachteten Crashstrukturen. Validiert wird die Methode an einer Robustheitsanalyse von crashrelevanten Komponenten. Dazu wird in einem Frontcrash die Position des Stoßfängers eines Fahrzeugs variiert. Die entstehenden Deformationsmodi werden anhand von linearen und nichtlinearen Dimensionsreduktionsalgorithmen visualisiert und identifiziert, wobei die nichtlinearen Methoden signifikant besser abgeschnitten haben, da sie die Deformationsmodi klarer herausgestellt haben. Ein großer Nachteil dieser Herangehensweise ist, dass sie netzabhängig arbeitet. Wenn verschiedene Vernetzungen verglichen werden sollen, muss entsprechend eine Referenzvernetzung eingeführt werden, um die FE-Daten darauf zu übertragen.

Kracker et al. (2020) analysieren ebenfalls lineare und nichtlineare Dimensionsreduktionsalgorithmen anhand der plastischen Dehnungen eines Längsträgers. Die Daten stammen dazu aus einer Robustheitsuntersuchung eines reduzierten FE-Modells des vorderen Fahrzeugteils und werden entsprechend der in Abschnitt 4.1 vorgestellten Projektion auf eine Kugeloberfläche und der Diskretisierung in Voxel aufbereitet. Durch die Betrachtung der plastischen Dehnungen bei der Dimensionsreduktion lässt sich auf das Verformungsverhalten der betrachteten Komponente schließen. Dieses Verformungsverhalten wird von Kracker et al. (2023) genutzt, um automatisiert Ausreißer aus einer Sammlung an FE-Simulationen zu detektieren. Das soll den/die Ingenieur*in bei der Analyse der Menge an FE-Daten unterstützen.

Bohn et al. (2013) clustern in einem ersten Schritt die Knotenverschiebungen eines Längsträgers in einem Crashlastfall über verschiedene Simulationen hinweg, sodass in einem Cluster Knoten mit ähnlichen Verschiebungen vorhanden sind. Die gefundenen Cluster werden dann an einen Algorithmus zur Dimensionsreduktion übergeben. Dadurch werden die Verschiebungen in einen niedrigdimensionalen Raum eingebettet. Weiterführende Untersuchungen zur Dimensionalität des Einbettungsraums werden in (Bohn et al. 2016) durchgeführt. Sowohl für das Clustering als auch für die Dimensionsreduktion wurden lineare und nichtlineare Methoden miteinander verglichen. Bohn et al. (2013), Garcke und Iza-Teran (2015) und Kracker et al. (2020) kommen schlussendlich zur Erkenntnis,

dass nichtlineare Methoden zur Dimensionsreduktion von Crashdaten besser geeignet sind gegenüber linearen Methoden. Letztendlich bleibt die Wahl der konkreten Methode aber stets problemspezifisch.

Von Hahner et al. (2020) wird vorgeschlagen, die Dimensionsreduktion basierend auf einem LSTM-Autoencoder durchzuführen. Ein LSTM-Autoencoder kann zeitabhängige Signale verarbeiten und in eine niedrigdimensionale Einbettung der originalen Daten überführen, hier einer 24-dimensionalen Einbettung. Die von Hahner et al. (2020) vorgeschlagene Methode beruht, wie bereits in Abschnitt 4.1 aufgezeigt wurde, auf Quadern, die die Geometrie der Struktur umschließen, den OBBs. Basierend auf der Einbettung der Positionen der OBBs lässt sich mit dieser Vorgehensweise eine Analyse der Deformationsmodi durchführen. Das hat den Vorteil, dass eine Simulation frühzeitig abgebrochen werden kann, falls ein unerwünschter Deformationsmodus vorhergesagt wird. Trotz der einfachen geometrischen Repräsentation der Strukturen können laut den Autoren unterschiedliche Deformationsverhalten identifiziert werden.

Zum Clustern des Deformationsverhaltens von primär eindimensional ausgeprägten Strukturen eignet sich eine von Diez (2019) bzw. Diez et al. (2016) entwickelte Methodik, die auf der geometrischen Dimensionsreduktion auf eine Regressionslinie aus Abschnitt 4.1 basiert. Das Clustering wird direkt auf den Regressionslinien verschiedener vergleichbarer Simulationen durchgeführt, um eine Auskunft über verschiedene Deformationsmodi zu erlangen.

4.2.3 Strukturoptimierung

Viele der in diesem Kapitel bisher vorgestellten Methoden können in Optimierungsmethoden integriert werden. In diesem Abschnitt hingegen liegt der Fokus auf Methoden, bei denen die KI essentieller Bestandteil der entwickelten Optimierungsmethode ist.

Kazi et al. (2022) stellen ein direktes Verfahren vor, welches mittels MLPs das optimale Querschnittsverhältnis zur Energieaufnahme von unterschiedlich belasteten Komposit-Profilstrukturen vorhersagt. Datengrundlage sind reale Experimente, welche unter quasistatischer Belastung durchgeführt wurden. Eingangsgrößen in das MLP sind verschiedene Lasten und Energiegrößen. Die Autoren erreichen mit diesem Vorgehen eine gute Vorhersagegüte.

Ein mit KI unterstützter Optimierungsalgorithmus in multikriteriellen Optimierungen wird von Gao et al. (2022) vorgestellt. Dazu wird ein NN mit radialen Basisfunktionen als Aktivierungsfunktion herangezogen. In einer ersten Iteration wird dieses Modell auf Datenbasis eines modifizierten LHS trainiert. Dadurch soll das globale Verhalten der Problemstellung durch die breite Abtastung des Entwurfsraums ermittelt werden.

Auf dem entstandenen Metamodell wird die Paretofront ermittelt. In diesem Bereich werden adaptiv weitere Punkte hinzugefügt, sodass die lokale Güte des Metamodells verbessert wird. Durch ein iteratives Wiederholen des Vorgehens passt sich die gefundene Paretofront immer weiter der wahren Paretofront an. Die Methode wird an einem Seitencrash erfolgreich validiert.

Liu et al. (2017) bzw. Liu et al. (2018) stellen eine Methode zur Optimierung von Materialphasen in Multimaterialstrukturen vor, welche durch Clusteringmethoden unterstützt wird. Die Optimierung von Multimaterialstrukturen erlaubt es eine auf die Optimierungsaufgabe zugeschnittene Verteilung von Materialbereichen mit individuellen Materialeigenschaften zu finden. Hergestellt werden solche Multimaterialstrukturen häufig durch additive Fertigung.

Die Methode teilt sich in drei Schritte auf:

1. Generierung eines konzeptuellen initialen Entwurfs, beispielsweise mit der HCA-Methode,
2. Clustern von Bereichen mit ähnlichen Strukturantworten im initialen Entwurf, um die Anzahl der elementweisen Entwurfsvariablen signifikant zu reduzieren und
3. Metamodellbasierte Optimierung der Materialphasen mit dem *Efficient Global Optimization*-Algorithmus (EGO-Algorithmus) (Jones et al. 1998), bei dem das Metamodell adaptiv durch Hinzufügen von Stützstellen angepasst wird.

Die Methode wurde u. a. an einem Längsträger im Fahrzeug validiert. Dabei wird die HCA-Methode genutzt, um die Blechdickenverteilung des Längsträgers zu optimieren. Der gefundene initiale Entwurf wird in vier Cluster aufgeteilt, welche dann mit dem EGO-Algorithmus bezüglich einer Maximierung der Energieaufnahme und einer Minimierung der maximal wirkenden Kraft auf den Längsträger optimiert wird. Nach den Autoren stellt diese Methode eine effiziente und effektive Möglichkeit bereit, hochgradig nichtlineare Crashproblemstellungen zu lösen.

Im Bereich der Topologieoptimierung hat Bujny (2020) eine auf einer niedrigdimensionalen Level-Set-Darstellung und evolutionären Algorithmen beruhende Methodik zur Optimierung von Crashstrukturen vorgestellt. Da dieser Ansatz durch eine hohe Anzahl an Funktionsaufrufen geprägt ist, wird dieser mit verschiedenen ML-Methoden erweitert, um mit weniger Funktionsaufrufen auszukommen. Dazu gehört die Vorhersage von Strukturantworten aus den Crashsimulationen, die Approximation von Sensitivitätsinformationen zur Nutzung in gradientenverstärkten, evolutionären Optimierungen (Bujny et al. 2016) und zur Vorhersage von topologischen Anpassungen.

4.2.4 Streuanalyse

In der Entwicklung von Crashstrukturen ist ein Verständnis des zugrundeliegenden, systematischen Rauschens in dem Simulationsmodell entscheidend. Herauszufinden, welche Variablen das Rauschen am stärksten beeinflussen, erlaubt dem/der Ingenieur*in eine direkte Beurteilung und Einschätzung der Streuung bezüglich ihrer Auswirkung auf die zu entwickelnde Komponenten. In der Literatur gibt es einige Ansätze, die sich damit beschäftigen, Streuquellen ausfindig zu machen und zu visualisieren. Häufig werden ML-Methoden dafür genutzt.

Die hier aufgeführten Quellen zeigen die theoretischen Grundlagen zur Streuanalyse, aus denen die kommerzielle Software DIFFCRASH (SIDACT GmbH 2024) hervorgegangen ist. Mit dieser Software lässt sich Modellrauschen durch physikalische Bifurkationen und numerische Instabilitäten aufdecken und Ursachen zuordnen. Dadurch kann der/die Ingenieur*in zielgerichtete Modifikationen am betrachteten Modell zur Reduktion der Streuung vornehmen.

Thole et al. (2010) zeigen eine Methode, um viele verschiedene Simulationsläufe eines Fahrzeugmodells miteinander zu vergleichen. Angewandt werden dazu die lineare PCA, um globale Unterschiede zwischen den Simulationsläufen zu identifizieren. Dabei werden verschiedene physikalische Effekte in einem Modus zusammengefasst, welcher die größte Varianz in den Daten beschreibt. Zusätzlich wird die Difference-PCA vorgestellt, welche sich dazu eignet einzelne Komponenten in einem Simulationslauf bezüglich der Korrelation zwischen der Streuung an verschiedenen Auswertestellen zu analysieren. Für die Difference-PCA werden die Auswertegrößen zunächst geclustert, um stark korrelierende Komponenten ermitteln zu können. Validiert wurden die verschiedenen Methoden an einem Datensatz aus Simulationen eines Fahrzeugmodells, bei dem Blechdicken im Fahrzeug variiert wurden. Eine weiterführende Studie an einem anderen Fahrzeugmodell ist in Borsotto et al. (2015) zu finden. Mertler (2022) stellt in seiner Dissertation eine Erweiterung der Methodik auf nichtlineare Dimensionsreduktionsalgorithmen vor.

5 Heuristikentwicklung mit Reinforcement Learning zur Strukturversteifung

5.1 Möglichkeiten des Machine Learnings zur Unterstützung der Graphen- und Heuristikbasierten Topologieoptimierung

Die GHT soll durch den Einsatz von KI erweitert und unterstützt werden. Daraus ergeben sich Fragestellungen, die im Folgenden beleuchtet werden. Ziel ist es, ein Konzept zur Integration von KI in die Optimierung crashbelasteter Strukturen mit der GHT zu entwickeln, dieses zu motivieren und zu begründen.

Welche Möglichkeiten zur Erweiterung der GHT gibt es? Zur Klärung dieser Frage muss zunächst geklärt werden, welche Schnittstellen zur GHT vorhanden sind. Damit sind alle Stellschrauben gemeint, die den Prozess der GHT ausmachen und angepasst, erweitert bzw. unterstützt werden können. Grundsätzlich ist der Einsatz von KI an jeder dieser Schnittstellen möglich, wenn auch mit unterschiedlichem ingenieurtechnischem Potential.

Zu den naheliegendsten Schnittstellen gehören der Optimierungsablauf der GHT, die zugrundeliegenden FE-Simulationen und die Heuristiken zur Topologieänderung. Aus diesen Schnittstellen lassen sich Ansätze formulieren, welche im Folgenden aufgezeigt werden:

- *Direkter Ansatz* – Bei diesem Ansatz greift die KI in den bestehenden Optimierungsablauf der GHT ein. Die KI kann dann ausgehend von aus Datensätzen gesammelten Erfahrungen Vorschläge für optimierte Entwürfe generieren. Die Vorhersage kann auf dem Lastfall oder den Simulationsergebnissen des initialen Entwurfs bzw. den Ergebnissen nach einer gewissen Anzahl an klassisch berechneten Iterationen beruhen.

Das Gute bei dieser Herangehensweise ist, dass die gesamte oder ein großer Teil der Optimierung übersprungen werden kann. Dabei würden aber im Mittel keine besseren Strukturen gefunden werden als mit der GHT ohne KI-Erweiterung. Denn alle Trainingsgrundlagen basieren auf dem aktuellen Stand der GHT und den beinhalteten Heuristiken. Die KI würde diese Zusammenhänge implizit anlernen, aber nicht verbessern oder neue Zusammenhänge lernen.

Ansätze, in denen optimierte Entwürfe vorhergesagt werden, stellen Sosnovik und Oseledets (2017), Zhang et al. (2019), Lei et al. (2019), Chandrasekhar und Suresh (2021) und Jeong et al. (2023) bereits für linear-statische Topologieoptimierungen vor, die mittels Methoden des MLs optimierte Entwürfe vorhersagen.

- *Simulationsorientierter Ansatz* – Aufgrund des Zeit- und Ressourcenaufwands von Crashberechnungen kann die GHT davon profitieren, wenn Rechenzeit bei jeder einzelnen benötigten FE-Simulation eingespart werden kann. Mit verschiedenen Methoden der KI können auf Basis einer angerechneten Crashsimulation die restlichen Deformationen prinzipiell vorhergesagt werden. Für eine allgemeingültige Anwendung müssen die Methoden aber noch weiterentwickelt werden, um detaillierter und möglichst problemunabhängig arbeiten zu können, sodass diese nicht nur für ein konkretes Bauteil antrainiert sind. Ebenso reicht es nicht aus, lediglich das Deformationsfeld vorherzusagen. Auch andere Feldgrößen und skalare Strukturantworten werden für die Strukturanalyse durch die Heuristiken der GHT benötigt. Wie auch bei dem o. g. direkten Ansatz wird mit dieser Herangehensweise die Effizienz der GHT gesteigert, aber nicht die Qualität der optimierten Strukturen.
- *Heuristikorientierte Ansätze* – Der Optimierungsablauf der GHT sieht vor, dass die besten Entwürfe aus der aktuellen Iteration in die nächste Iteration übergeben werden. Hier kann die KI eingreifen und die geeignetste Heuristik vorhersagen, sodass nur der aus dieser Heuristik abgeleitete Entwurf als Grundlage in der nächsten Iteration genutzt wird. Eine Heuristik ist dann geeignet, wenn sie in einer Referenzoptimierung ohne KI aktiviert worden wäre und zur Generierung der optimierten Struktur direkt beitragen würde. So können viele unnötige Heuristikaufrufe inklusive zugehöriger FE-Simulationen und somit Ressourcen eingespart werden. Weiterhin lebt die GHT von ihrem Potential zur Erkundung des Entwurfsraums. Sollte die KI also einen suboptimalen Heuristikvorschlag generieren, ist das Risiko groß, dass die GHT eine schlechtere finale Struktur findet, da die Erkundung des Entwurfsraums stark eingeschränkt wird.

  Alternativ kann die KI als weitere, konkurrierende Heuristik angesehen werden, die eigene Topologievorschläge generiert. Diese Herangehensweise hat, unter Verwendung geeigneter Methoden des MLs, das Potential, dass die GHT bessere Strukturen findet, die ohne der Verwendung einer KI nicht gefunden worden wären. Dadurch erhöht sich allerdings der Ressourcenaufwand und der Zeitaufwand der Optimierung.

Welche Methoden der KI eignen sich zur Unterstützung der GHT? Grundsätzlich können viele Methoden des MLs zur Erweiterung der GHT herangezogen werden, wobei Methoden des SLs am leichtesten zu integrieren sind. Der Trainingsdatensatz besteht dann aus vielen einzelnen GHT-Optimierungen. Die Eingangsgrößen in das ML-Modell sind geometrische und mechanische Eigenschaften der betrachteten Struktur und die Ausgangsgröße ergibt sich aus dem betrachteten Ansatz. Für den direkten Ansatz sind das Vorhersagen von optimierten Strukturentwürfen, für den simulationsorientierten Ansatz sind das skalare Größen und Feldgrößen der Simulationen und im heuristikorientierten Ansatz zur Heuristikvorhersage ist es die geeignetste Heuristik als Grundlage für die nächste Iteration. Für die o. g. Ansätze eignen sich aufgrund der Problemkomplexität u. a. MLPs, für Informationen mit strukturierter Nachbarschaftsbeziehung CNNs, für zeitabhängige Daten LSTMs und aufgrund der Geometriebeschreibung durch Graphen GNNs.

Zum Finden neuartiger Topologievorschläge könnten theoretisch auch Methoden des SLs herangezogen werden, wobei dann die Generierung des Datensatzes unabhängig von GHT-Optimierungen erfolgen sollte, um neue Erkenntnisse anlernen zu können und nicht lediglich das von der GHT bekannte Wissen anzulernen. Folglich müssen sinnvolle Topologien z. B. aufwendig über einen Zufallsgenerator bestimmt werden. Das ML-Modell soll nicht auf Topologien trainiert werden, die für die untersuchte Zielfunktion und das gegebene strukturmechanische Verhalten ungünstig sind. Entsprechend fallen bei der Generierung eines solchen Datensatzes viele Daten an, die nicht genutzt werden, sodass der Ansatz höchst sample-ineffizient ist.

Stattdessen bieten sich die Methoden des RLs an. Durch das Trial-and-Error-Prinzip der Methode ist es möglich aus Topologievorschlägen zu lernen, die im Sinne der Zielfunktion, suboptimal oder gar schlecht sind. Ein weiterer Vorteil besteht darin, dass die optimale Struktur vor dem Training nicht bekannt sein muss. Durch das RL erlernt das ML-Modell selbstständig, was sinnvolle Topologien sind und was nicht und wie diese generiert werden. Der Nachteil am RL ist allerdings die naturgemäß hohe Anzahl an benötigten Funktionsaufrufen, um das ML-Modell zu trainieren. Der Input in das Modell ist dann eine Beobachtung des aktuellen mechanischen Zustands und die Ausgabe ist ein Vorschlag zur Topologieänderung. Ein RL-basierter Ansatz zur Topologieoptimierung linear-statisch belasteter Strukturen wurde von Hayashi und Ohsaki (2020) und Hayashi und Ohsaki (2022) bereits vorgestellt.

Eine mögliche evolutionsbasierte Herangehensweise ist der *NeuroEvolution of Augmented Topologies*-Algorithmus (NEAT-Algorithmus) (Stanley und Miikkulainen 2002). Bei diesem Algorithmus werden die Topologie und die Gewichte eines MLPs durch *Mutation*, *Selektion* und *Rekombination* evolviert. Basierend auf einer *Fitness-Funktion* wird bemessen, wie gut die mit dem aktuell vorliegenden MLP generierte Struktur ist, wodurch das Training getrieben wird. Typisch für evolutionäre Ansätze ist die besonders hohe Anzahl an benötigten Funktionsaufrufen. Gefundene Strategien können aufgrund des

Funktionsprinzips des Algorithmus eine gute Performance und ggfs. sogar bessere Performance als mit RL-Algorithmen erreichen. Diese Vermutung wird von Taylor et al. (2006) exemplarisch bestätigt. Aufgrund der noch höheren Anzahl benötigter Funktionsaufrufe gegenüber dem RL, wird diese Herangehensweise in dieser Arbeit nicht weiterverfolgt.

Welche Art von Erweiterung ist sinnvoll für die GHT? Alle der genannten Ansätze können zumindest theoretisch herangezogen werden, um die GHT durch KI zu erweitern. Die meisten Ansätze scheitern aber an dem zu geringen Verhältnis aus Zeit- bzw. Ressourcenaufwand und Nutzen.

Entsprechend wird der *direkte Ansatz* zur Vorhersage von optimierten Entwürfen nicht weiterverfolgt. Motiviert wird das durch einen Vergleich zwischen den linear-statischen Methoden u. a. von Zhang et al. (2019) und der Komplexitätssteigerung durch die Betrachtung von Crashproblemstellungen. Die Nichtlinearitäten, Verzweigungspunkte und das Rauschen in der Crashmechanik erhöhen die Komplexität der Entwurfsvorhersage deutlich. Neben der intrinsischen Schwierigkeit müsste dann zusätzlich auch der Datensatz entsprechend groß gewählt werden. Zhang et al. (2019) nutzen zur Entwicklung des ML-Modells der linearen Problemstellung bereits 80 000 Datenpunkte. Für Crashsimulationen ist das in dieser Form, in der das ML-Modell an die konkrete Aufgabenstellung geknüpft ist, nicht praktikabel.

Der *simulationsorientierte Ansatz* wird ebenfalls nicht weiterverfolgt. Dieser Ansatz bringt große Unsicherheiten in die Optimierung ein, wodurch die Qualität des optimierten Entwurfs leidet. Ebenso ist dieser Ansatz auf Basis des aktuellen Forschungsstandes nicht flexibel genug, für unterschiedlichste Modelle und Lastfälle die Simulationsergebnisse für skalare Größen bzw. Feldgrößen vorherzusagen. Auch die Generierung benötigter Trainingsdaten ist in diesem Rahmen unpraktikabel.

Zur methodischen Weiterentwicklung der GHT ist der *heuristikorientierte Ansatz* zur Findung neuer Topologien durch eine KI-Heuristik am vielversprechendsten. Dadurch kann die GHT den Entwurfsraum besser abtasten und somit auch bessere Strukturen finden. Entsprechend liegt der Fokus dieser Dissertation auf der Generierung einer neuen KI-basierten Heuristik.

Wie bereits angesprochen, kann dies durch verschiedene Methoden des ML erreicht werden. Obwohl RL-Methoden bekannt für ihre hohe Anzahl an benötigten Funktionsaufrufen sind, überwiegen die bereits genannten Vorteile insbesondere gegenüber Methoden des SLs und dem NEAT-Algorithmus. Die zu erwartende hohe Anzahl an Funktionsaufrufen und die damit verbundene hohe Rechenzeit bei RL-Methoden ist nur bei der Erstellung der Heuristik gefordert und nicht mehr beim Einsatz der Heuristik im Prozess der Strukturoptimierung mit der GHT.

5.2 Anforderungen an die Heuristik

Damit die zu entwickelnde Heuristik möglichst sinnvoll in die GHT integriert werden kann, muss diese einige Anforderungen erfüllen. Das Verbessern der Zielfunktion der Optimierungsaufgabe ist das primäre Ziel der Heuristik. Auslegungskriterien in der Crashoptimierung sind in verschiedenen Optimierungsaufgaben häufig konträr. Entsprechend ist es schwierig eine Heuristik zu entwickeln, die mehrere konträre Optimierungsziele gleichzeitig bedienen kann. Stattdessen erfüllt jede Heuristik einen konkreten Zweck und die GHT filtert entsprechend die geeignetsten Heuristiken in jeder Iteration heraus. Die neu zu entwickelnde Heuristik soll folglich auch einen konkreten Zweck erfüllen, die das Finden einer optimierten Struktur bezüglich eines konkreten Optimierungsziels in der Crashoptimierung unterstützt. Aus dem Prinzip der konkurrierenden Heuristiken folgt, dass es zwar erstrebenswert, aber nicht zwingend nötig ist, dass die Heuristik stets bestmögliche Topologien vorschlägt.

Funktionale Restriktionen müssen nicht direkt von der Heuristik berücksichtigt werden. Die Einhaltung der Restriktionen wird durch die GHT überprüft und wird, sofern vom Benutzer aktiviert, durch die folgende Dimensionierung und Formoptimierung eingehalten. Die Erfüllung der Herstellungsrestriktionen muss durch die Heuristik bestmöglich gewährleistet werden. Wenn die vorgeschlagene Form und Topologie des Entwurfs die Herstellungsrestriktionen nicht erfüllt, ist der Entwurf hinfällig und die neue Heuristik trägt nicht zum Finden einer besseren Struktur bei.

5.3 Funktionsprinzip der Heuristik

Im Folgenden wird das Funktionsprinzip der neuen Heuristik konkretisiert. Der RL-Agent soll iterativ über verschiedene Schritte hinweg Wände in die Struktur einbringen, um so die Steifigkeit dieser zu erhöhen. Aufgrund der intrinsischen Komplexität von crashmechanischen Optimierungen ist das Ziel eine möglichst einfache Beschreibung der mechanischen Größen zu ermöglichen. Ebenso soll die Beschreibung in verschiedenen Modellen, Lastfällen und Optimierungen konsistent sein, damit der RL-Agent eine Chance hat, die mechanischen Eigenschaften zu analysieren und entsprechend zu handeln.

Als naheliegende Ansätze sind hier GNNs, GCNs und der graph2vec-Algorithmus (Narayanan et al. 2017) zu nennen. Bei den beiden zuerst genannten Ansätzen handelt es sich, wie bereits in Abschnitt 3.3 angesprochen wurde, um eine Form von NNs, welche beliebige Graphdaten flexibel verarbeiten können. Diese NNs könnten dann den aktuellen Strukturgraphen mitsamt zugehöriger mechanisch relevanter Informationen entgegennehmen und entweder einen Einbettungsvektor generieren oder direkt einen neuen Graphen mit neuer Topologie vorhersagen.

Der graph2vec-Algorithmus kann ebenfalls einen Graphen entgegennehmen und daraus einen Einbettungsvektor generieren, welcher den Graphen beschreibt. Der Algorithmus wird dem UL zugeschrieben und kann Graphen beliebiger Größe aufgabenunabhängig einbetten. Der Einbettungsvektor kann dann mit klassischen Feed-Forward- bzw. MLP-Architekturen verwendet werden. Das MLP könnte dann beispielsweise eine Klassifizierungsaufgabe lösen und bestimmen, welche Kante in den Strukturgraphen eingebracht werden kann.

In dieser Dissertation werden diese Ansätze trotz der vielversprechenden Möglichkeiten nicht verfolgt. Motiviert wird dies auf Basis der folgenden Argumente (Trilling et al. 2024):

- Es ist nicht geklärt, wie eine skalare oder vektorielle Aktion eines Agenten in eine graphunabhängige, konsistente Beschreibung einer in den Strukturgraphen einzubringenden Kante übersetzt werden kann. Daraus folgt auch, dass dieser zu klärende Zusammenhang durch den Agenten aufwendig angelernt werden muss, was die Trainingskomplexität weiter steigert.
- Die genannten graphbasierten Algorithmen unterliegen einer direkten Abhängigkeit zu den Trainingsdaten, auf denen die ML-Modelle trainiert werden. Um geeignete Einbettungen für beliebige Strukturgraphen generieren zu können, müssten auch FE-Modelle über das Rahmenmodell hinaus untersucht werden, um eine Anwendbarkeit für beliebige FE-Modelle in der GHT zu ermöglichen.
- Der Fokus dieser Dissertation liegt auf der mechanischen Anwendbarkeit der entwickelten RL-Heuristik. Zum Zeitpunkt der Entwicklung der Methode war kein verwendbarer öffentlicher Code bekannt, der eine Schnittstelle zwischen Graphen und den GNN- oder graph2vec-basierten Agenten bzw. der zugrundeliegenden Umgebung umsetzt.
- Die Trainingskomplexität ist in einem Ansatz, bei dem der gesamte Strukturgraph mitsamt mechanischem Verhalten als Entscheidungsgrundlage genutzt wird, naturgemäß groß. Der Agent müsste Muster in dem strukturmechanischen Verhalten auf Basis der GNNs erkennen und darauf basierend seine Entscheidungen durchführen. Bei unbegrenzt vielen Modell- und Lastfallmöglichkeiten und daraus resultierenden Deformationsverhalten erscheint das äußerst schwierig.

Deshalb werden in dieser Dissertation Teilstrukturen des Graphen betrachtet, die sog. Zellen. Exemplarisch wird eine Zelle in Abbildung 5–1 dargestellt. Durch die Betrachtung lokaler Zellen reduziert sich die mechanische Komplexität des Systems. Gleichzeitig erlauben die Zellen eine konsistente und strukturierte Beschreibung über beliebige Strukturgraphen hinweg. Aufgrund der Umsetzung dieser konsistenten Beschreibung kann ein Agent nur Zellen, die von einer fixen Anzahl von Vertices aufgespannt werden, handhaben. Deshalb werden verschiedene Agenten für verschiedene Zelltypen trainiert. Im Folgenden werden die Zellen in Abhängigkeit ihrer Anzahl an Zellseiten n_{Zs} als n_{Zs}-Zellen bezeichnet.

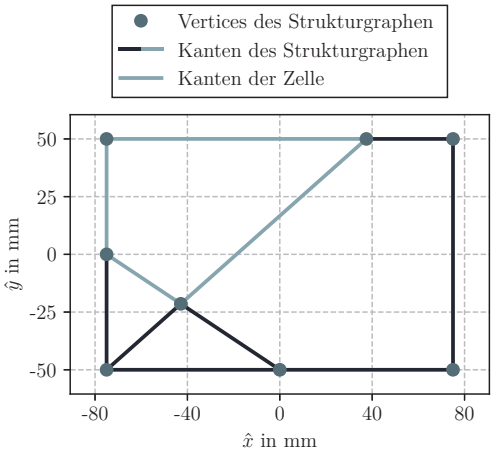


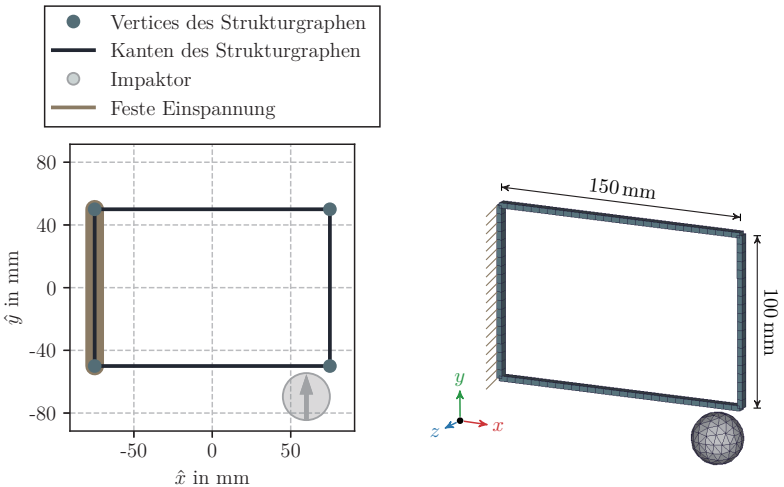
Abbildung 5–1: Darstellung einer 4er-Zelle in einem Beispielstrukturgraphen, die durch vier Vertices aufgespannt wird

Ziel des Agenten soll es sein, die Steifigkeit einer Zelle zu maximieren, wodurch die Steifigkeit der gesamten Struktur steigen soll. Steife Crashstrukturen sind beispielsweise dann gefordert, wenn der verfügbare Deformationsweg gering ist, wie bei dem Batteriegehäuse in Elektrofahrzeugen. Ausgehend von einer leeren Zelle schlägt der Agent dann iterativ Kanten vor, die in die lokale Zelle des Strukturgraphen eingebracht werden, um die Steifigkeit der Zelle bestmöglich zu erhöhen. So können auch mehrere Kanten je Heuristikaufruf in die Zelle eingebracht werden. Die Strukturmasse wird bei Änderungen der Form und Topologie konstant gehalten, indem die Blechdicke der Wände des extrudierten Profils skaliert werden. Durch Hinzufügen einer Kante in den Graphen werden also die den Kanten zugewiesenen Wände dünner. Die Entscheidungsbasis, welche Kante zur Steifigkeitsmaximierung in die Zelle eingebracht werden soll, basiert auf geometrischen und mechanischen Eigenschaften der im Crash deformierten und undeformierten Zelle.

Die Betrachtung von lokalen Zellen hat Grenzen. Zum einen ist der Einfluss auf die Zielfunktion der Optimierungsaufgabe eingeschränkt. Ebenso entspricht die Zielfunktion zur Optimierung der Zelle i. Allg. nicht der Zielfunktion der eigentlichen Optimierungsaufgabe. Da die Zielfunktion für Heuristiken unbekannt ist, gilt das für alle in der GHT verwendeten Heuristiken. Zum anderen kann die lokale Veränderung der Struktur die Gesamtperformance der Struktur verschlechtern. Dann hat die GHT aber stets die Möglichkeit auf die bereits etablierten Heuristiken zurückzugreifen.

5.4 Wahl des Simulationsmodells

Zur Generierung von Daten für das Trainieren des Agenten werden zufallsgenerierte Simulationsmodelle und Lastfälle aufgebaut, um eine möglichst große Diversität in der geometrischen Beschreibung und in den Strukturantworten der Zellen zu generieren. Die zufallsbasierten Modelle sind Derivate eines zeit- und ressourceneffizienten Referenz-Rahmenmodells. Eine frühe Version des Rahmenmodells inklusive zugehörigem Lastfall wurde von Ortmann und Schumacher (2013) vorgestellt und in Ortmann (2015) modifiziert. Der das Rahmenmodell beschreibende Strukturgraph wird in Abbildung 5–2 zusammen mit dem zugehörigen Lastfall dargestellt.



(a) Strukturgraph des Rahmens in Profilkoordinaten **(b)** FE-Modell des Rahmens

Abbildung 5–2: Strukturgraph des Referenz-Rahmenmodells, welcher mittels GRAMB in das zugehörige FE-Modell übersetzt wird

Die Struktur besteht aus Aluminium, welches durch ein isotropes, elastoplastisches Materialmodell abgebildet wird. Der Elastizitätsmodul beträgt 70 000 N/mm^2, die Dichte ist 2,7 g/cm^3 und die Querkontraktionszahl weist einen Wert von 0,33 auf. Tabelle 5–1 zeigt die Fließspannung in Abhängigkeit der effektiven plastischen Dehnung.

Tabelle 5–1: Fließspannung in Abhängigkeit der effektiven plastischen Dehnung für Aluminium

Effektive plastische Dehnung	0,0	0,02	0,04	0,06	0,08	0,1	0,2	0,5
Fließspannung in N/mm^2	240,0	265,0	285,0	300,0	310,0	315,0	335,0	340,0

Bei einer Wandstärke von 3 mm ergibt sich eine Masse von 20,25 g. Der Strukturquerschnitt wird 5 mm orthogonal zum Querschnitt extrudiert. Diese geringe Extrusionstiefe erlaubt eine weitere Verringerung der Simulationszeit, um die hohe Anzahl an benötigten Funktionsaufrufen beim Trainieren des Agenten zu kompensieren. Bei der gewählten Elementgröße von 5 mm wird die Extrusionstiefe durch eine einzelne Elementreihe an vollintegrierten Schalenelementen abgebildet. Die Schalenelemente besitzen 5 Integrationspunkte in Dickenrichtung. Der Impaktor ist eine starre Kugel mit einer Masse von 105,1 g und einer initialen Geschwindigkeit von 10 m/s. Die Kugel hat einen Durchmesser von 30 mm.

Das Simulationsmodell hat durch die starke Vereinfachung Grenzen bezüglich des physikalischen Verhaltens und der Verallgemeinerungsfähigkeit des Agenten, der auf den durch das Modell generierten Daten trainiert. Starke lokale Krümmungen der Struktur können aufgrund der groben Vernetzung nicht sauber abgebildet werden. Die geringe Extrusionstiefe der Struktur verhindert, dass diese beulen kann. Entsprechend können solche Phänomene nicht abgebildet und antrainiert werden. Dennoch können die Wände der Struktur knicken. Der Impaktor ist stets eine starre Kugel mit gleichen Ausmaßen, die andere Deformationsmodi der Zelle erzeugt, als es andere Impaktoren, wie beispielsweise eine starre Wand, erzeugen würden. Das Material der Struktur wird für das Training des Agenten nicht verändert.

5.5 Implementierung der Trainingsumgebung

5.5.1 Aufbau der Umgebung

Die Trainingsumgebung ist in zwei zentrale Module aufgeteilt (Trilling et al. 2024). Das *Reset*-Modul ist dafür da, einen neuen Zufallsgraphen zu generieren, einen randomisierten Lastfall aufzubauen und eine valide Zelle in diesem Graphen auszuwählen, die als Trainingsgrundlage für die aktuelle Episode genutzt wird. Daraus abgeleitet wird dann die Observation, welche dem Agenten als Entscheidungsgrundlage dient, um eine Kante auszuwählen, die in den Graphen eingezogen werden soll. Das *Step*-Modul ist Teil einer Schleife, welche iterativ vom Agenten vorgeschlagene Kanten in den Graphen einbringt und aus der Simulation des abgeleiteten Modells neue Observationen, Rewards und eine Terminierungs-Flagge generiert. Die Terminierungs-Flagge dient zur Bestimmung, ob die Episode terminiert wird und kann durch den Agenten, aber auch durch die Umgebung gesetzt werden. Der prinzipielle Ablauf einer Episode der Trainingsumgebung wird in Abbildung 5–3 dargestellt. Eine Episode umfasst dabei alle Schritte von der Generierung der Zelle bis hin zur finalen Zelle.

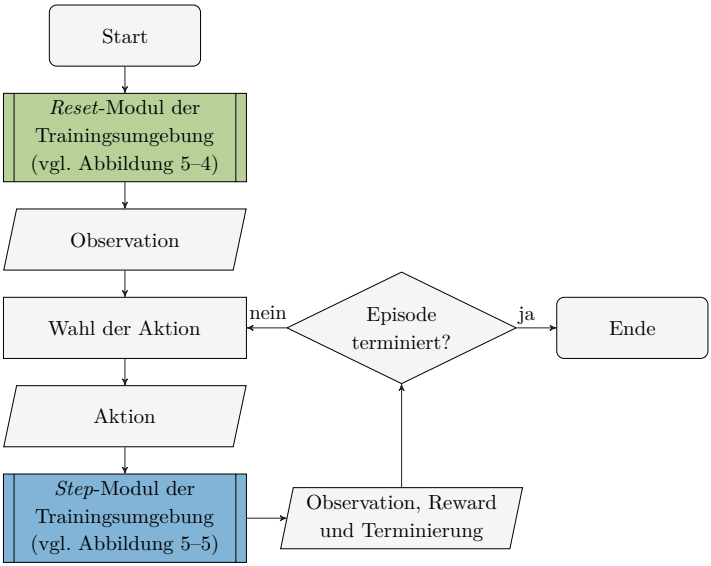


Abbildung 5–3: Ablauf einer Episode in der Trainingsumgebung (modifiziert aus Trilling et al. (2024))

Das Reset-Modul und das Step-Modul sind Subprozesse, deren grober Ablauf hier als Übersicht strukturiert aufgezeigt wird. Für eine detaillierte Beschreibung der Teilschritte des Ablaufs dienen die Abschnitte 5.5.2 bis 5.5.11.

Der Ablauf des in Abbildung 5–4 dargestellten Reset-Moduls beginnt mit der Generierung eines herstellbaren Zufallsgraphen, welcher die mechanische Grundlage für die vom Agenten zu lösende Optimierungsaufgabe darstellt. Zusätzlich wird ein randomisierter Lastfall generiert, welcher die Einspannung des aus dem Zufallsgraphen abgeleiteten Modells und die Position und den Geschwindigkeitsvektor des Impaktors beinhaltet.

Anschließend werden alle möglichen Zellen im Graphen identifiziert. Warum hier alle Zellen betrachtet werden, der Agent schließlich aber nur auf einer Zelle trainiert, wird in den nächsten Abschnitten motiviert. Die detektierten Zellen werden in dem Graphen hinterlegt. Dabei werden die Kanten jeder Zelle mittig mit einem neuen Vertex geteilt, um diversere Topologien durch den Agenten vorschlagen zu können. Neue Kanten können zwischen den Vertices der Zelle eingezogen werden. Da sich durch diesen Prozess die Kantenlängen verkürzen, muss erneut überprüft werden, ob der Graph bzw. das daraus abgeleitete Modell fertigbar ist. Ist er das nicht, wird der Graph verworfen und ein neuer Zufallsgraph generiert.

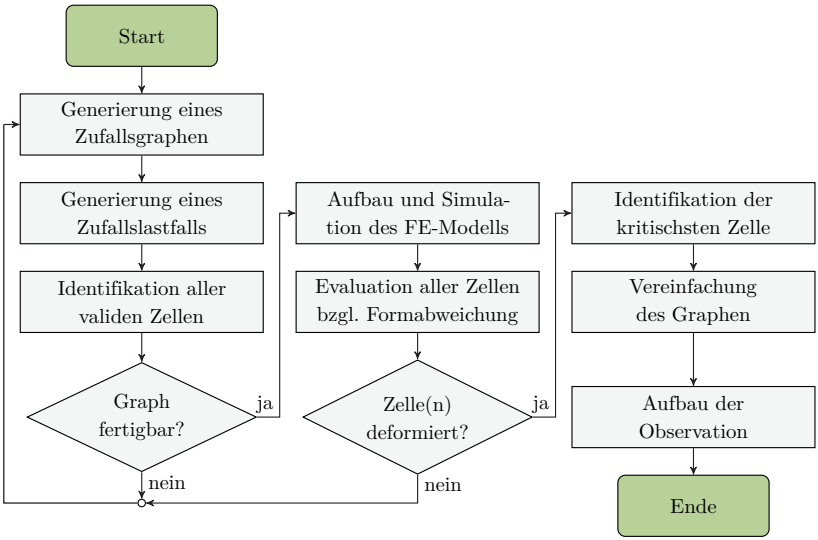


Abbildung 5–4: Ablaufdiagramm des Reset-Moduls (modifiziert aus Trilling et al. (2022a))

Aus dem ermittelten Graphen wird das FE-Modell abgeleitet und simuliert. Dadurch, dass mehrere Zellen betrachtet werden, wird jede einzelne Zelle hinsichtlich ihrer Steifigkeit bewertet. Dafür wird ein Formabweichungsmaß eingeführt, welches später detailliert eingeführt wird. Mindestens eine der Zellen muss sich ausreichend deformiert haben. Ansonsten wird das Modell verworfen und der Reset-Prozess beginnt von vorne, wodurch ein neuer Zufallsgraph generiert wird. Aus der Menge an deformierten Zellen wird dann die kritischste ausgewählt. Es muss also stets eine FE-Simulation durchgeführt werden, um zu validieren, dass mindestens eine Zelle überhaupt deformiert wurde. Ein Training auf einer zu wenig deformierten Zelle soll dadurch ausgeschlossen werden. Weil der Graph noch die geteilten Kanten aller möglichen Zellen beinhaltet, wird der Graph vereinfacht und alle Vertices aus der Kantenteilung der nicht weiter betrachteten Zellen wieder entfernt.

Final wird die Observation aus mechanischen und geometrischen Größen aufgebaut. Wie das genau geschieht, wird später beschrieben. Diese Observation wird dem Agenten übergeben und dient dann als Entscheidungsgrundlage, welche Kante als nächstes in die leere Zelle eingezogen werden soll, um langfristig, d. h. unter Berücksichtigung weiterer eingebrachter Kanten, die Deformation der Zelle zu minimieren und dadurch ihre Steifigkeit zu maximieren.

Das in Abbildung 5–5 dargestellte Step-Modul wird in einer Schleife nach dem Reset-Modul aufgerufen. Das Simulationsmodell und die Zelle sind bereits generiert und auf Basis der abgeleiteten Observation bestimmt der Agent eine Aktion. Die Aktion beinhaltet die Kante, die in die Zelle eingebracht werden soll. Das Einziehen der Kanten zur Änderung der Topologie der Zelle geschieht im ersten Schritt des Step-Moduls. Im gleichen Schritt wird auch die Blechdicke aller Wände des Simulationsmodells so skaliert, dass die Masse des Modells konstant bleibt.

Anschließend wird das FE-Modell aufgebaut, simuliert und die zu optimierende Zelle mit ihrer neuen Topologie evaluiert. Diese Teilschritte funktionieren weitgehend identisch, wie die aus dem Reset-Modul.

Abschließend werden der Reward zur Bewertung des Nutzens der neu eingebrachten Kante und die Terminierungs-Flagge, welche Auskunft gibt, ob die Episode beendet wird, bestimmt. Ebenso wird analog zum Reset-Modul eine neue Observation aufgebaut, welche als Entscheidungsgrundlage für die nächste einzuziehende Kante dient.

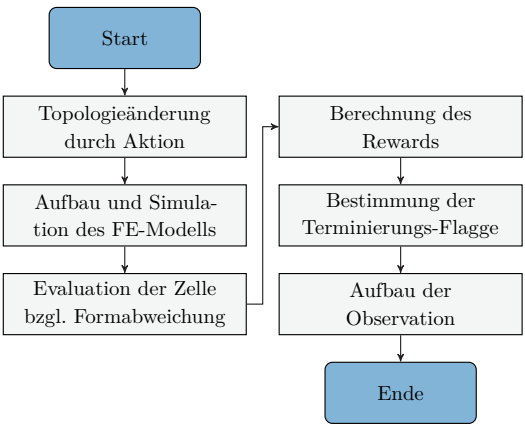


Abbildung 5–5: Ablaufdiagramm des Step-Moduls (modifiziert aus Trilling et al. (2022a))

Die Crashsimulationen in dieser Dissertation werden mit LS-DYNA (Livermore Software Technology Corporation (LSTC) 2023a) durchgeführt und der Code für die Agenten und die zugrundeliegende Umgebung ist in PYTHON geschrieben. Zur Anwendung kommen primär die Bibliotheken GYM (Brockman et al. 2016) und STABLE-BASELINES3 (Raffin et al. 2021). GYM ist eine Bibliothek, welche eine konsistente und standardisierte Möglichkeit zur Implementierung von Umgebungen für RL bietet. STABLE-BASELINES3 verfügt über verschiedenste RL-Algorithmen, welche mit den Umgebungen aus GYM interagieren können. So können Agenten flexibel auf unterschiedlichen Umgebungen trainiert werden. Die Implementierung der Graphstrukturen der GHT für die Umgebung wird mit der Bibliothek NETWORKX (Hagberg et al. 2008) umgesetzt. Für numerische

Operationen wird auf die Bibliothek NUMPY (Harris et al. 2020) zurückgegriffen. Die als Binärdaten gespeicherten Simulationsergebnisse aus LS-DYNA werden mit der Bibliothek QD (Diez 2018) ausgelesen.

5.5.2 Graphbasierte Datenstrukturen

Bevor die einzelnen Teilschritte der Trainingsumgebung detailliert erklärt werden können, wird zunächst auf die der Umgebung zugrundeliegenden Datenstrukturen eingegangen. Diese Datenstrukturen dienen als Grundlage zur Datenverarbeitung in den Teilschritten. Es handelt sich um verschiedene Repräsentationen der zu optimierenden Struktur und der Zelle in Form von Graphen. Dazu gehören der Strukturgraph G_S, der Zellengraph G_Z, der modifizierte Zellengraph G_Zm und der Evaluationsgraph G_E. Diese Graphen werden im Folgenden erläutert. Die Menge der Vertices eines Graphen G wird mit $\mathcal{V}^{\{G\}}$ und die Menge der Kanten des Graphen G mit $\mathcal{E}^{\{G\}}$ angegeben.

Bereits bekannt ist der *Strukturgraph* G_S. Der in diesem und in weiteren Abschnitten genutzte Strukturgraph (vgl. Abbildung 5–1) wird in Abbildung 5–6 mit zwei in die Zelle eingebrachten Kanten dargestellt.

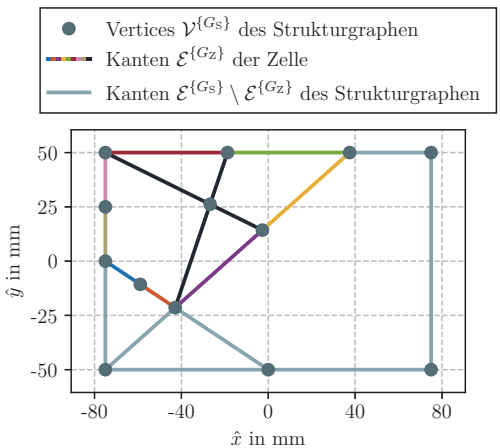


Abbildung 5–6: Beispiel eines Strukturgraphen G_S mit hervorgehobener Zelle und zwei eingezogenen Kanten

Da sich die zwei eingebrachten Kanten in einem Schnittpunkt schneiden, existiert ein Vertex im Schnittpunkt, der den Schnitt auflöst. Dadurch besteht das Zellinnere aus effektiv vier Kanten. Der Zellrahmen selbst ist, wie bereits angesprochen wurde, je Zellseite mittig durch einen weiteren Vertex geteilt, um eine größere Form- und Topologiediversität zu ermöglichen.

Bisher wurde von Zellen als abstraktes Konzept gesprochen. Programmintern sind Zellen eine eigene graphbasierte Datenstruktur, ähnlich wie der Strukturgraph, aber um essentielle Funktionen erweitert. Diese Struktur wird *Zellengraph* G_Z oder kurz Zelle genannt. Der Zellengraph besteht ausschließlich aus im Raum positionierten Vertices, die Bestandteil der Zelle sind und den Kanten, die die Vertices der Zelle miteinander verbinden. Ein Beispiel eines Zellengraphen wird in Abbildung 5–7 gegeben.

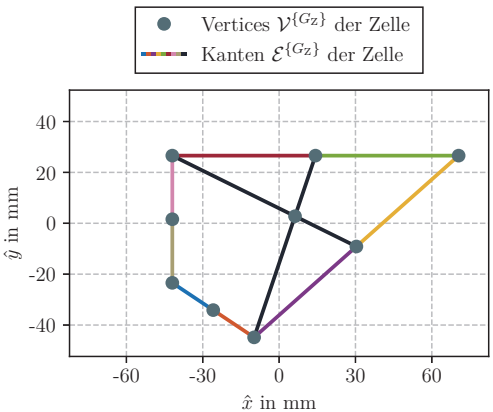
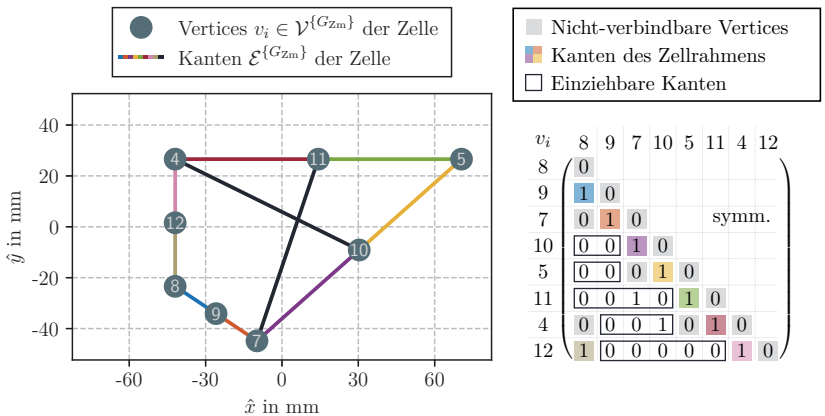


Abbildung 5–7: Beispiel eines Zellengraphen G_Z mit zwei eingezogenen Kanten

Funktionalitäten, die der Zellengraph gegenüber dem Strukturgraphen zusätzlich hat, sind u. a. Checks, ob die Zelle geometrische Kriterien einer validen Zelle erfüllt. Besonders entscheidend ist die Strukturierung der Kanten im Zellengraphen, weshalb diese farblich zwischen Zellen konsistent gekennzeichnet sind. Begonnen wird mit der Kante, die am weitesten unten links in der Zelle positioniert ist. Diese wird blau eingefärbt. Gegen den Uhrzeigersinn werden dann stets in gleicher Reihenfolge die folgenden Kanten eingefärbt. Das soll hervorheben, dass gleich eingefärbte Kanten des Zellrahmens für den Agenten in der Observation immer an der identischen Stelle hinterlegt sind. Dieser Aufbau der Zelle ermöglicht das Generieren von strukturierten Daten und wurde bereits von Trilling et al. (2022a) und Trilling et al. (2022b) eingeführt.

Das Konzept ist auf Zellen beliebiger Seitenzahl übertragbar und basiert auf der Adjazenzmatrix einer modifizierten Variante des Zellengraphen. Eine Adjazenzmatrix gibt an, welche Vertices in einem Graphen miteinander verbunden sind. Der generelle Aufbau

der Adjazenzmatrix ist für alle Zellen identisch, sofern die Vertices auf dem Rahmen der Zelle stets gleich sortiert sind. Als Konvention werden die Vertices analog zu den Kanten entlang des Zellrahmens entgegen des Uhrzeigersinns sortiert. Begonnen wird mit dem Vertex, der in der Zelle am weitesten unten links positioniert ist. Daraus folgt, dass Kanten in der Adjazenzmatrix über verschiedene Zellen hinweg, unabhängig von der Vertexbezeichnung, stets an der gleichen Stelle stehen. Das ist die zentrale Idee, die es ermöglicht Observationen zu generieren, die zwischen verschiedensten Zellen konsistent sind. Hier ist es allerdings entscheidend, dass stets eine modifizierte, nicht-planare Variante eines Zellengraphen G_{Zm} herangezogen wird. Nicht-planar bedeutet in diesem Zusammenhang, dass sich Kanten des Graphen schneiden dürfen. Während bei dem bereits vorgestellten Zellengraphen G_{Z} Schnitte von neu eingebrachten Kanten innerhalb der Zelle durch das Einbringen neuer Vertices an den Schnittpunkten aufgelöst werden, werden die Schnitte in G_{Zm} nicht aufgelöst. So ist beim Einbringen einer neuen Kante stets ein Rahmenknoten mit einem anderen Rahmenknoten der Zelle verbunden. Nur dadurch ist eine konsistente Beschreibung verschiedenster Zellen mit gleicher Seitenanzahl gewährleistet. Das Beispiel aus Abbildung 5–8 zeigt einen solchen modifizierten Zellengraphen G_{Zm}, welcher aus dem Zellengraphen G_{Z} aus Abbildung 5–7 abgeleitet ist, mit sich schneidenden Kanten und der zugehörigen Adjazenzmatrix.



(a) Modifizierter Zellengraph G_{Zm}

(b) Adjazenzmatrix von G_{Zm}

$$([0\ 0]\ [0\ 0]\ [0\ 0\ 1\ 0]\ [0\ 0\ 1]\ [0\ 0\ 0\ 0\ 0])^{\mathsf{T}}$$

(c) Abgeleiteter Adjazenzvektor

Abbildung 5–8: Adjazenzmatrix und abgeleiteter Adjazenzvektor einer 4er-Zelle auf Basis einer modifizierten Variante des Zellengraphen G_{Zm}, bei dem sich eingebrachte Kanten schneiden (modifiziert aus Trilling et al. (2022a))

Strukturantworten können stets nach diesem Verfahren den Kanten zugeordnet werden. Ebenso ermöglicht genau dieses Vorgehen auch, dass alle „gleichen“ Kanten konsistent über verschiedene Zellen hinweg durch die Zuordnung einer eineindeutigen Kennziffer beschrieben werden können. Das ist relevant für den Agenten, der so durch eine Ganzzahl die einzuziehende Kante beschreiben kann.

Dadurch, dass die Zelle ein ungerichteter Graph ist, ist die Adjazenzmatrix stets symmetrisch. Ebenso ist der Rahmen der Zelle stets miteinander verbunden und nicht alle Vertices auf dem Zellrahmen lassen sich verbinden. Nicht verbinden lassen sich die Vertices, die vor dem Teilungsprozess direkt miteinander verbunden waren. So lässt sich die Matrix kompakter formulieren und durch reihenweise Konkatenation der noch freien Einträge in einen Vektor überführen. Dieser sortierte Vektor wird in dieser Dissertation *Adjazenzvektor* der jeweiligen Zelle genannt und stellt eine konsistente Beschreibung von Kanten verschiedener Zellen sicher. Er ist zusätzlich zur Adjazenzmatrix in Abbildung 5–8c dargestellt.

Basierend auf der Anzahl an Seiten einer Zelle lassen sich verschiedene Eigenschaften einer modifizierten Zelle quantifizieren. Diese Eigenschaften definieren u. a. die Größe des Aktions- und Observationsraums und sind damit für das Training eines Agenten entscheidend.

Die Anzahl der einziehbaren Kanten $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|$ in eine leere Zelle mit $|\mathcal{S}^{\{G_{\mathrm{Zm}}\}}|$ Seiten und der Anzahl an Vertices entlang des Rahmens der Zelle $|\mathcal{V}^{\{G_{\mathrm{Zm}}\}}|$ und damit folglich die Anzahl der Einträge des Adjazenzvektors lassen sich über den Zusammenhang

$$|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}| = \binom{|\mathcal{V}^{\{G_{\mathrm{Zm}}\}}|}{2} - \overbrace{\left(|\mathcal{V}^{\{G_{\mathrm{Zm}}\}}| + |\mathcal{S}^{\{G_{\mathrm{Zm}}\}}|\right)}^{\text{verbundene Zellkanten}} \tag{5–1}$$

berechnen. Die Anzahl aller einziehbaren Kanten entspricht dem Ziehen von zwei Vertices aus den Vertices $\mathcal{V}^{\{G_{\mathrm{Zm}}\}}$ entlang des Zellrahmens. Alle durch den Zellrahmen stets verbundenen Vertexkombinationen müssen abgezogen werden, da der Agent keinen Einfluss auf diese Kanten hat. Das sind Kanten an Vertices entlang des Rahmens, die direkt miteinander verbunden sind und die Vertices, welche ein Kantensegment aufspannen. Ein Kantensegment ist eine Menge an Kanten in einem Graphen, die miteinander über Vertices verbunden sind und die gleiche Orientierung aufweisen.

Die Anzahl aller Kanten einer Zelle $|\mathcal{E}_{\mathrm{ges}}^{\{G_{\mathrm{Zm}}\}}|$ ergibt sich aus der Summe der Kanten entlang des Rahmens $|\partial\mathcal{E}^{\{G_{\mathrm{Zm}}\}}|$ und der Anzahl der einziehbaren Kanten $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|$:

$$|\mathcal{E}_{\mathrm{ges}}^{\{G_{\mathrm{Zm}}\}}| = |\partial\mathcal{E}^{\{G_{\mathrm{Zm}}\}}| + |\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|. \tag{5–2}$$

Tabelle 5–2 stellt Werte von $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|$, $|\partial\mathcal{E}^{\{G_{\mathrm{Zm}}\}}|$ und $|\mathcal{E}_{\mathrm{ges}}^{\{G_{\mathrm{Zm}}\}}|$ in Abhängigkeit der Anzahl an Zellenseiten $|\mathcal{S}^{\{G_{\mathrm{Zm}}\}}|$ übersichtlich dar. Es ist zu erkennen, dass für eine steigende Anzahl an Zellenseiten $|\mathcal{S}^{\{G_{\mathrm{Zm}}\}}|$ die Anzahl an einziehbaren Kanten in die Zelle $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|$ schnell steigt. Somit steigt auch die Größe des Aktions- und Observationsraums schnell an.

Tabelle 5–2: Werte für die Anzahl einziehbarer Kanten $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|$, die Anzahl der Kanten entlang des Zellrahmens $|\partial\mathcal{E}^{\{G_{\mathrm{Zm}}\}}|$ nach dem Kantenteilungsprozess und die Gesamtanzahl aller Kanten $|\mathcal{E}_{\mathrm{ges}}^{\{G_{\mathrm{Zm}}\}}|$ in Abhängigkeit der Anzahl der Seiten einer Zelle $|\mathcal{S}^{\{G_{\mathrm{Zm}}\}}|$

$\lvert\mathcal{S}^{\{G_{\mathrm{Zm}}\}}\rvert$	$\lvert\mathcal{I}^{\{G_{\mathrm{Zm}}\}}\rvert$	$\lvert\partial\mathcal{E}^{\{G_{\mathrm{Zm}}\}}\rvert$	$\lvert\mathcal{E}_{\mathrm{ges}}^{\{G_{\mathrm{Zm}}\}}\rvert$
3	6	6	12
4	16	8	24
5	30	10	40

Der Evaluationsgraph $G_{\mathrm{E}} = G_{\mathrm{E}}(t)$, welcher in Abbildung 5–9 exemplarisch gezeigt wird, besitzt die gleichen Eigenschaften des Zellengraphen, wird aber so erweitert, dass Simulationsergebnisse zeitabhängig als Vertex- und Kantenattribute abgespeichert werden können. Die Kanten des Zellengraphen werden ersetzt durch viele einzelne Kanten und Vertices, welche aus den FE-Knoten und FE-Elementen in einem von dem/der Nutzer*in vorgegebenen Querschnitt entlang der Extrusionsachse des Simulationsmodells abgeleitet werden. Dadurch kann der Evaluationsgraph die Deformation der Zelle in einem Querschnitt des Profils repräsentieren.

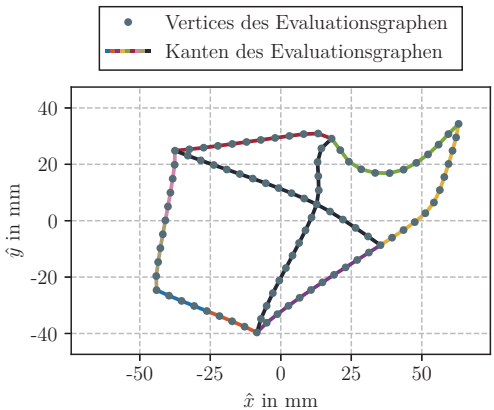


Abbildung 5–9: Beispiel eines Evaluationsgraphen $G_{\mathrm{E}}(t_{\mathrm{max}}^{\mathrm{IE}})$ zum Zeitpunkt $t_{\mathrm{max}}^{\mathrm{IE}}$, an dem die innere Energie der Zelle maximal ist

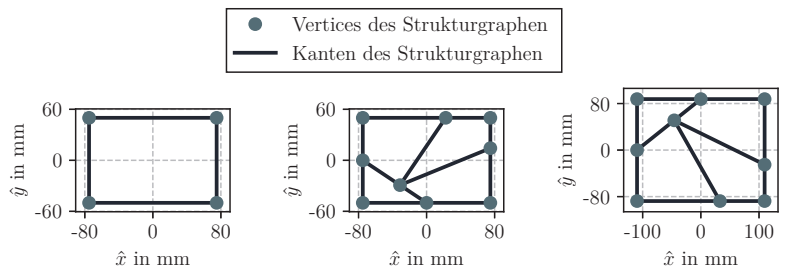
5.5.3 Generierung von Zufallsgraphen

Zur Generierung von Zufallsgraphen werden aus ca. 3000 dieser Arbeit vorangegangenen GHT-Optimierungen auf Basis des vorgestellten Rahmenmodells (vgl. Abbildung 5–2) 29 278 Strukturgraphen mit unterschiedlichen Topologien und Strukturanordnungen ex-

trahiert. Die Optimierungsaufgabe der vorangegangenen Optimierungen bestand darin, die maximale Beschleunigung des Impaktors unter Einhaltung von Verschiebungsrestriktionen zu minimieren. Dafür wurde die Impaktorposition und der Auftreffwinkel des Impaktors variiert.

Im Trainingsprozess des Agenten wird aus dieser Menge an Strukturgraphen ein Zufallsgraph gewählt, welcher zusätzlich noch weiter randomisiert wird. Der Vorteil in der Nutzung bestehender Graphen liegt darin, dass der Agent auf Strukturen und Strukturantworten trainiert, die aus realen GHT-Optimierungen stammen und damit eine anwendungsnahe Verteilung an Strukturen und Strukturantworten die Datengrundlage für das Training bildet. Das Ziel ist viele unterschiedliche Graphen zu erzeugen, die dann in der Simulation diverse mechanische Eigenschaften und Strukturantworten vorweisen. Dadurch soll der Agent verschiedenste Strukturverhalten durch die Observationen anlernen und erkennen.

Der zufällig gewählte Graph wird in seinen beiden Ausdehnungsrichtungen zusätzlich und unabhängig voneinander mit Faktoren zwischen 0,75 und 2 skaliert, welche entsprechend einer Gleichverteilung zufällig gewählt werden. Die Graphen werden ebenso gleichverteilt entlang der lokalen Profilachsen gespiegelt, um die Varianz in der Zellengeometrie weiter zu vergrößern. Ebenso wird die Blechdicke, die allen Wänden der extrudierten Struktur zugeordnet wird, zufällig zwischen 1 mm und 5 mm gewählt. Die Blechdicke ist als Attribut einer Kante Bestandteil des Graphen. Die Wahl der Wertebereiche des Skalierungsfaktors und der Blechdicke sind unter Beachtung ingenieurtechnischer Plausibilität so gewählt, dass die Zellengeometrie und Blechdicke aus typischen Aufgabenstellungen abgebildet wird. Durch die Anpassung der Größe des Strukturgraphen und der Blechdicke variieren die Massen zwischen verschiedenen, initialen Entwürfen. Die Schritte zur Randomisierung des dem Referenz-Rahmenmodell zugrundeliegenden Graphen wird in Abbildung 5–10 abgebildet.



(a) Referenzgraph **(b)** Zufällige Wahl der Topologie **(c)** Zufällige Bestimmung der Form und Dimension

Abbildung 5–10: Prozess zur Generierung von Zufallsgraphen basierend auf dem Graphen des Referenz-Rahmenmodells

Die Generierung zufallsbasierter Graphen und den daraus abgeleiteten Strukturen kann dazu führen, dass Strukturen vorgeschlagen werden, welche aus technischer Sicht nicht herstellbar sind. Dafür besitzt die GHT die Möglichkeit Fertigungsrestriktionen in die Optimierung zu integrieren und generierte Entwürfe zu überwachen und ggfs. auszusortieren. Die für den Trainingsprozess gewählten Fertigungsrestriktionen sind in Tabelle 5–3 dargestellt. Diese Restriktionen sind aus einer GHT-Optimierung des Referenz-Rahmenmodells von Ortmann (2015) abgeleitet und werden auf die Datengenerierung für den Trainingsprozess zugeschnitten. Falls ein zufallsgenerierter Graph die Fertigungsrestriktionen im Reset-Modul nicht erfüllt, wird dieser verworfen und es wird ein neuer Graph aufgebaut. Für den Fall, dass der Graph durch die Topologiemodifikation im Step-Modul nicht mehr hergestellt werden kann, wird die Episode terminiert, da die Herstellbarkeit durch das Einbringen weiterer Kanten nicht wiederhergestellt werden kann.

Tabelle 5–3: Fertigungsrestriktionen zur Datengenerierung für den Trainingsprozess, wobei $\mathcal{E}^{\{G_\mathrm{S}\}}$ die Menge aller Kanten des Strukturgraphen G_S und $\mathcal{V}^{\{e_i\}}$ die Menge der Vertices einer Kante e_i ist

Parameter	Restriktionsgrenze	Geltungsbereich
Kantenlänge $l_{e_i}^{\{G_\mathrm{S}\}}$	$l_{e_i}^{\{G_\mathrm{S}\}} \geq 10\,\mathrm{mm}$	$\forall e_i \in \mathcal{E}^{\{G_\mathrm{S}\}}$
Wandstärke $t_{e_i}^{\{G_\mathrm{S}\}}$	$1\,\mathrm{mm} \leq t_{e_i}^{\{G_\mathrm{S}\}} \leq 5\,\mathrm{mm}$	$\forall e_i \in \mathcal{E}^{\{G_\mathrm{S}\}}$
Abstand $d_{e_i,e_j}^{\{G_\mathrm{S}\}}$	$d_{e_i,e_j}^{\{G_\mathrm{S}\}} \geq 5\,\mathrm{mm}$	$\forall e_i, e_j \in \mathcal{E}^{\{G_\mathrm{S}\}}$ mit $e_i \neq e_j \wedge \mathcal{V}^{\{e_i\}} \cap \mathcal{V}^{\{e_j\}} = \emptyset$
Winkel $\alpha_{e_i,e_j}^{\{G_\mathrm{S}\}}$	$\alpha_{e_i,e_j}^{\{G_\mathrm{S}\}} \geq 15^\circ$	$\forall e_i, e_j \in \mathcal{E}^{\{G_\mathrm{S}\}}$ mit $e_i \neq e_j \wedge \mathcal{V}^{\{e_i\}} \cap \mathcal{V}^{\{e_j\}} \neq \emptyset$

5.5.4 Generierung von Zufallslastfällen

Für die Generierung des Lastfalls wird ein zufälliges Kantensegment des zugrundeliegenden Graphen ausgewählt, entlang dessen die starre Kugel als Impaktor positioniert wird. Das gewählte Kantensegment muss dabei eine Mindestlänge von dem fünffachen Impaktorradius aufweisen – das entspricht 75 mm –, damit die Auftrefffläche groß genug ist, die Struktur die Energie des Impaktors vollständig aufnehmen kann und der Impaktor nicht von der Struktur abrutscht.

Für eine weitere essentielle Eigenschaft eines zur Positionierung des Impaktors in Frage kommenden Kantensegments wird das Konzept des Randes eines Graphen eingeführt. Die Randvertices $\partial\mathcal{V}^{\{G\}}$ eines Graphen G entsprechen einer Menge an Vertices, die die konvexe Hülle basierend auf den Koordinaten der Graphvertices aufspannen. Für die Kanten entlang des Randes eines Graphen $\partial\mathcal{E}^{\{G\}}$ gilt, dass diese ausschließlich durch Vertices aus $\partial\mathcal{V}^{\{G\}}$ aufgespannt sein dürfen.

Entsprechend muss das zur Auswahl stehende Kantensegment vollständig auf dem Rand $\partial\mathcal{E}^{\{G_S\}}$ des Graphen liegen, damit der Impaktor initial außerhalb der Struktur positioniert werden kann. Das ist für nicht-konvexe Graphen ansonsten nicht sicher gegeben. Ein Beispiel zur Visualisierung der Randkanten eines Strukturgraphen G_S ist in Abbildung 5–11 gegeben.

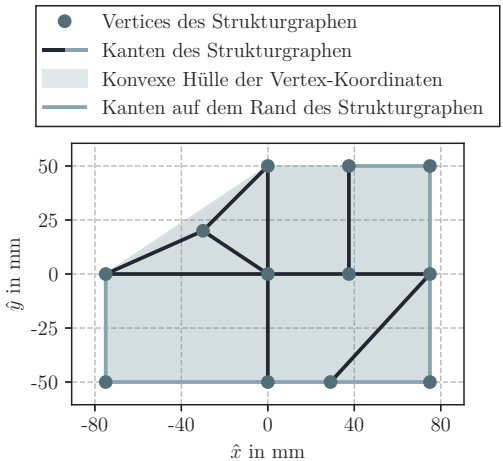


Abbildung 5–11: Bestimmung der Kanten $\partial\mathcal{E}^{\{G_S\}}$ auf dem Rand eines nicht-konvexen Strukturgraphen G_S auf Basis der Koordinaten der Vertices $\mathcal{V}^{\{G_S\}}$

Die Darstellung zeigt, dass manche Kantensegmente zwar auf dem Äußeren der Struktur liegen, aber sich nicht auf dem Rand der konvexen Hülle befinden. Rein theoretisch ist es sogar möglich einen beliebigen Graphen zu konstruieren, bei dem kein einziges Kantensegment auf dem Rand der konvexen Hülle liegt. Ein solches Phänomen tritt bei den zugrundeliegenden, aus der GHT abgeleiteten Topologien nicht auf. Entsprechend wird dieser Umstand vernachlässigt.

Zur Generierung einer Impaktorposition für den Lastfall wird ein zufälliges Kantensegment gewählt, das durch eine Teilmenge der Randkanten $\partial\mathcal{E}^{\{G_S\}}$ des Strukturgraphen aufgespannt wird. Entlang des Kantensegments wird der Impaktor mit einem Offset von 3 mm vom Impaktorrand zum Kantensegment unter Beachtung der Blechdicke zufällig positioniert. Die Seite des Kantensegments, auf die der Impaktor positioniert wird, ergibt sich aus dem Vorzeichen des Skalarprodukts zwischen dem Richtungsvektor der Kugel zur gewählten Position auf dem Kantensegment und dem Richtungsvektor zwischen Kugel und Schwerpunkt der Struktur. Potentiell könnte ein benachbartes Kantensegment

fest eingespannt sein. Der Impaktor soll mit diesem Bereich nicht kollidieren. Zu den Rändern des Segments hin gibt es deshalb einen Offset von je 25 mm zum Mittelpunkt des Impaktors, in dem der Impaktor nicht positioniert werden darf.

Die initiale Flugrichtung des Impaktors wird ebenfalls zufällig bestimmt. Entweder fliegt der Impaktor senkrecht auf das gewählte Kantensegment oder der Impaktor fliegt in Richtung des Schwerpunkts des Strukturgraphen. Dadurch soll eine noch größere Diversität in der Strukturmechanik erreicht werden. Die initiale Geschwindigkeit des Impaktors wird zufällig aus einer Normalverteilung $\mathcal{N}(\mu_{v_0}, \sigma_{v_0}^2)$ bestimmt. Der Erwartungswert der Geschwindigkeit μ_{v_0} bestimmt sich aus einer linearen Abhängigkeit zur Wandstärke $t_{e_i}^{\{G_S\}}$ des Strukturgraphen über

$$\mu_{v_0} = \frac{v_{\max} - v_{\min}}{t_{\max} - t_{\min}} \cdot t_{e_i}^{\{G_S\}} + \frac{t_{\max} \cdot v_{\min} - t_{\min} \cdot v_{\max}}{t_{\max} - t_{\min}}. \tag{5–3}$$

Dabei ist $v_{\min} = 10\,\mathrm{m/s}$, $v_{\max} = 25\,\mathrm{m/s}$, $t_{\min} = 1\,\mathrm{mm}$ und $t_{\max} = 5\,\mathrm{mm}$. Hier muss deutlich angemerkt werden, dass das Ziel der linearen Abhängigkeit der Geschwindigkeit und der Wandstärke eine äußerst grobe Abschätzung zwischen der zu absorbierenden Energie und der Struktursteifigkeit ist. Ein linearer Zusammenhang ist mechanisch nicht korrekt, ist aber einfach umzusetzen und reicht für die Generierung des Lastfalls aus. Ohnehin ist die konkrete Wirkung der Impaktorgeschwindigkeit auf die Zelle ohne zusätzliche Informationen schwierig abzuschätzen. Die maximale Geschwindigkeit $v_{\max} = 25\,\mathrm{m/s}$ wird hoch gewählt, da die Geschwindigkeit bevorzugt überschätzt werden soll. Dadurch wird die Wahrscheinlichkeit erhöht, eine deformierte Zelle bei der initialen Simulation im Reset-Modul zu finden.

Die final gewählte, initiale Geschwindigkeit v_0 folgt aus einem Sample v_0' aus der Normalverteilung mit $\sigma_{v_0} = 3\,\mathrm{m/s}$, welches mithilfe der Clip-Funktion $\mathrm{clip}\,(x; a; b) := \min(\max(x, a), b)$ auf den zulässigen Bereich zwischen $v_{\min}$ und $v_{\max}$ limitiert wird:

$$v_0 = \mathrm{clip}\left(v_0' \sim \mathcal{N}(\mu_{v_0}, \sigma_{v_0}^2); v_{\min}; v_{\max}\right). \tag{5–4}$$

Die Wahl des Kantensegments zur Positionierung der Einspannung funktioniert weitestgehend analog zu der Bestimmung des Kantensegments zur Positionierung des Impaktors. Die Mindestlänge der eingespannten Kante muss länger sein als die Hälfte der längsten Kante im Graphen, um eine großflächige, aber dennoch variabel lange Einspannung zu ermöglichen. Es wird zusätzlich sichergestellt, dass das gewählte Kantensegment ein anderes ist, als das, was für den Impaktor genutzt wurde. Für die bei dem Modellaufbau mit GRAMB entstehenden FE-Knoten entlang des Kantensegments sind alle translatorischen und rotatorischen *Degrees of Freedom* (DoF) (deutsch: *Freiheitsgrade*) gesperrt, was einer festen Einspannung entspricht.

In Abbildung 5–12 wird die Positionierung des Impaktors für ein gewähltes Kantensegment und die Positionierung der festen Einspannung an einem anderen Kantensegment visualisiert.

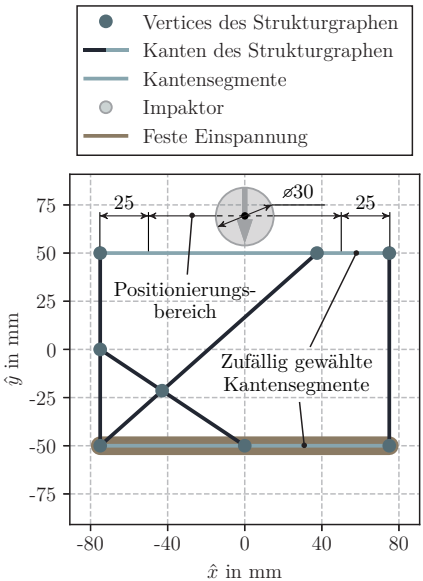


Abbildung 5–12: Exemplarische Positionierung des Impaktors mit Positionierungsbereich und exemplarische Positionierung der festen Einspannung

5.5.5 Identifikation valider Zellen

Das Finden valider Zellen in einem Strukturgraphen basiert auf der Identifikation aller Zyklen in einem Graphen. Ein Zyklus in einem Graphen ist ein geschlossener Pfad entlang von Vertices, in denen kein Vertex mehrfach vorkommt. Dafür wird der von Johnson (1975) entwickelte und von Hagberg et al. (2008) implementierte Algorithmus für gerichtete Graphen genutzt. Der Strukturgraph selbst ist ein ungerichteter Graph, kann aber problemlos in einen gerichteten Graphen überführt werden, indem die Vertices einer ungerichteten Kante durch zwei gerichtete Kanten in beide Richtungen miteinander verbunden werden.

Schon für wenig komplexe Graphen ist die Anzahl aller gefundenen Zyklen hoch. Alle diese Zyklen spannen potentiell eine Zelle auf und werden deshalb auf ihre Eignung für eine Zelle hin nach den folgenden Kriterien überprüft.

- Die Anzahl der Vertices auf dem Zellrahmen muss konstant sein für einen Agenten. Nur so ist gewährleistet, dass die Observationen und Aktionen den korrekten Kanten zugeordnet werden können. Zu einer 4er-Zelle gehören demnach, aufgrund des Teilungsprozesses an jeder der $|\mathcal{S}^{\{G_{\mathrm{Zm}}\}}| = 4$ Seiten, stets $|\mathcal{V}^{\{G_{\mathrm{Zm}}\}}| = 8$ Vertices entlang des Rahmens.

- Die Zelle muss konvex sein. So ist sichergestellt, dass die eingezogene Kante vollständig innerhalb der Zelle liegt und den Zellrahmen nicht durchdringt.
- Innerhalb der von einem Zyklus aufgespannten Zelle befinden sich häufig Vertices und Kanten des Strukturgraphen. Damit die Zelle valide ist, muss diese leer sein, d. h., dass sich keine Vertices oder Kanten initial in der Zelle befinden.
- Um einen Mindesteinfluss der Zelle auf das Strukturverhalten zu gewährleisten und um zu kleine Kanten im Graphen, die durch den Kantenteilungsprozess entstehen könnten, zu vermeiden, muss die von der Zelle aufgespannte Fläche mindestens 5 % von der durch die konvexe Hülle aufgespannte Querschnittsfläche des Strukturgraphen ausmachen.
- Die bisher genannten Kriterien sind geometrische Kriterien. Zusätzlich dazu muss die initiale Zelle auch ein Mindestmaß an Deformation und somit absorbierter Energie aufweisen, damit eine Optimierung der Zelle durch den Agenten bzgl. der Formabweichung der Zelle überhaupt Sinn ergibt. Wie das Formabweichungsmaß berechnet wird, wird später genauer beschrieben.

Es ist möglich, dass eine 4er-Zelle bereits auf einer oder mehreren Seiten durch einen mittig platzierten Vertex geteilt wurde, wie das Beispiel in Abbildung 5–13 zeigt. An diesem Vertex ist dann eine Kante des Strukturgraphen von außerhalb der Zelle angebunden. Auch solche Zyklen werden detektiert und als Kandidat für eine Zelle herangezogen. Dann werden in dem Kantenteilungsprozess nur die verbleibenden Kanten des Zellrahmens geteilt. Solche Zellen werden in dieser Dissertation Zellen höherer Ordnung genannt. Naturgemäß sind solche teilenden Vertices auf dem Zellrahmen durch die äußere Struktur nicht exakt mittig, so wie es bei den „künstlich“ eingezogenen Vertices der Fall ist. Das ist aber nicht kritisch, solange solche Vertices nicht zu weit von der Mitte abweichen, was in der Praxis in GHT-basierten Entwürfen i. d. R. nicht eintritt.

Theoretisch ist es auch möglich, eine originale Kante des Strukturgraphen durch mehrere statt nur einem mittig platzierten Vertex zu teilen. So ist die Freiheit bzgl. der Topologie noch größer. Das wird in dieser Dissertation allerdings nicht verfolgt, da das Einbringen mehrerer Vertices entlang einer originalen Kante in vielen deutlich kürzeren Kanten resultiert, welche dann die Fertigungsrestriktionen der GHT oft nicht mehr erfüllen.

Wie bereits angesprochen wurde, werden im Reset-Modul alle möglichen Zellen identifiziert. Für jede dieser Zellen wird iterativ der Kantenteilungsprozess durchgeführt. Durch das Ändern des Graphen im Kantenteilungsprozess einer Zelle kann es dazu kommen, dass eine der folgenden Zellen nicht mehr valide ist, da die Vertexzahl des Zellrahmens nicht mehr mit der geforderten Vertexzahl übereinstimmt. Dieses Problem ist mit der vorliegenden Implementierung auch nicht zu umgehen. Es lässt sich allerdings abschwächen, indem eine Präferenz für die Zellen vorgenommen wird. Der einfachste Weg wäre es, dass das Reset-Modul viele verschiedene Simulationen durchführt und identifiziert wird, welche Zelle die kritischste ist. Das ist aber nicht effizient, denn so könnte man von vorneherein alle Zellen isoliert betrachten und müsste nicht verschiedene Zellen in

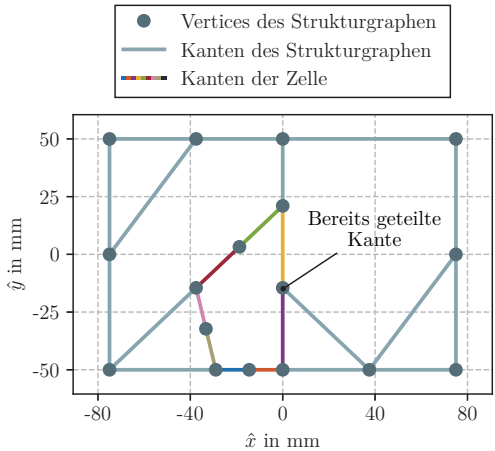


Abbildung 5–13: Beispielzelle höherer Ordnung in einem Strukturgraphen, bei der ein Kantensegment durch die äußere Struktur bereits geteilt ist und nicht zusätzlich durch den Kantenteilungsprozess geteilt werden muss

einen Graphen integrieren. Entsprechend kann kein simulationsbasiertes Maß für die Wichtigkeit der Zelle herangezogen werden. Stellvertretend wird die aufgespannte Fläche der Zelle genutzt. Je größer die Zelle, desto wichtiger wird sie, da der Einfluss auf die Gesamtstruktur steigt. Entsprechend werden die größten Zellen zuerst geteilt. Zusätzlich sind größere Zellen weniger anfällig hinsichtlich einer Verletzung der Fertigungsrestriktionen nach dem Kantenteilungsprozess, bei dem naturgemäß nicht herstellbare, zu kurze Kanten entstehen können.

5.5.6 Aufbau und Simulation des Modells

Die Ableitung des Simulationsmodells aus dem Strukturgraphen wurde bereits detailliert in Abschnitt 2.5.2 besprochen. Wichtig ist im Rahmen der Trainingsumgebung anzumerken, dass GRAMB das interne Vernetzungstool der GHT benutzt. Der Vorteil liegt in der Geschwindigkeit, mit der das Simulationsmodell vollständig aufgebaut werden kann. Bei der geforderten hohen Anzahl an Funktionsaufrufen kann somit eine Menge Zeit gegenüber proprietären Vernetzungstools gespart werden, welche ggfs. unnötig Zeit durch Programmstarts und Lizenzabfragen in Anspruch nehmen.

5.5.7 Evaluationsmaß für Strukturperformance

Mit dieser Dissertation wird ein Formabweichungsmaß eingeführt, das die „Steifigkeit“ der Struktur darüber beschreibt, wie stark diese sich in einem wirkenden und für das Maß unbekannten Lastzustand deformiert. Das Maß ist entsprechend vollständig deformations- bzw. verschiebungsbasiert und ist daher kein Steifigkeitsmaß im engeren Sinne. Es gilt stets für den aktuell herrschenden Lastzustand auf die Zelle. Je weniger sich die Zelle deformiert, desto steifer ist sie für den gegebenen Lastzustand. Dadurch, dass das Maß lastunabhängig agiert, kann die Umgebung über die Deformation automatisch die für die jeweils vorliegende Lastverteilung kritischste Zelle ausmachen. Diese sich stark deformierenden Zellen sind die Zellen, die aus ingenieurtechnischer Betrachtung bezüglich ihrer Steifigkeit am sinnvollsten zu optimieren sind. Dadurch, dass das Maß vollständig auf Verschiebungen basiert, unterliegt es keinem nennenswerten Rauschen, wie es bei Schnittkräften oder anderen crashrelevanten Strukturantworten der Fall ist. Das vereinfacht das Training für den Agenten.

Zur Bestimmung des Formabweichungsmaßes wird der Evaluationsgraph G_E herangezogen. Dieser beinhaltet eine Repräsentation des deformierten Querschnitts der Zelle zu jedem Zeitpunkt der Simulation. Die undeformierte Zelle wird mit einer deformierten Kopie der Zelle in deren Schwerpunkten überlagert. Das wird in Abbildung 5–14 gezeigt. Der Zeitpunkt, an dem die Deformation der Zelle ermittelt wird, ist der Auswertezeitpunkt $t^\mathrm{IE}_\mathrm{max}$. Dieser wird so gewählt, dass die innere Energie der Zelle maximal ist.

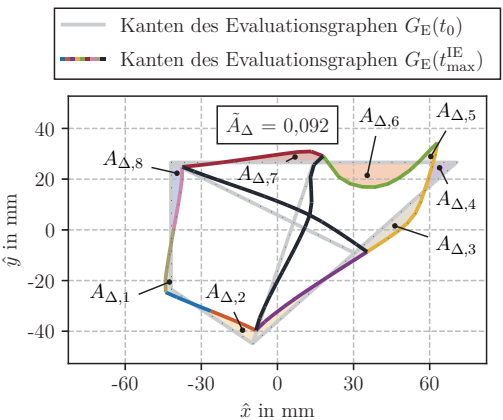


Abbildung 5–14: Visualisierung des Formabweichungsmaßes auf Basis der Überlagerung zweier Instanzen des Evaluationsgraphen $G_\mathrm{E}(t)$ zu unterschiedlichen Zeitpunkten

Durch die Überlagerung ergeben sich Differenzflächen $A_{\Delta,j}\big|_t$, welche die Abweichung von der deformierten Zelle zur undeformierten Zelle ausgewertet zum Zeitpunkt t beschreiben. Das Formabweichungsmaß $\tilde{A}_\Delta$ wird aus den Differenzflächen mit der Formel

$$\tilde{A}_\Delta = \frac{\sum A_{\Delta,j}\big|_{t^{\mathrm{IE}}_{\max}}}{A\big|_{t_0} + A\big|_{t^{\mathrm{IE}}_{\max}}}. \tag{5–5}$$

berechnet. Zur Normierung des Maßes auf einen Wertebereich zwischen 0 und 1 werden die Querschnittsflächen der deformierten und undeformierten Zelle $A\big|_{t_0}$ und $A\big|_{t^{\mathrm{IE}}_{\max}}$ herangezogen. Die Summe aus beiden Werten wird gewählt, da die Differenzflächen nach „innen“ und nach „außen“ von dem Evaluationsgraphen berechnet werden und somit ansonsten doppelt gezählt würden.

Ein Wert von 0 bedeutet, dass die deformierte und undeformierte Struktur exakt aufeinander liegen. Somit ergeben sich keine Differenzflächen und der Zähler in Gleichung (5–5) wird 0, sodass der gesamte Bruch 0 wird. Ein Wert von 1 bedeutet, dass die deformierte Zelle keine Querschnittsfläche mehr aufspannt. Dafür muss die Zelle in eine mathematisch niedrigere Dimension kollabieren, also zu einem Punkt oder einer Linie. Das ist ein theoretischer Wert, der so in der Praxis nicht erreicht wird. Werte von $\tilde{A}_\Delta \lesssim 0{,}8$ hingegen sind auch praktisch möglich.

Eine wichtige Ergänzung zur Berechnung des Formabweichungsmaßes $\tilde{A}_\Delta$ ist, dass die Starrkörperrotation der Zelle herausgerechnet werden muss, um zu vermeiden, dass Differenzflächen entstehen und bewertet werden, die keinen Beitrag zur Deformation der Struktur leisten. Dazu wird der mittlere euklidische Abstand zwischen den mit der Rotationsmatrix $\boldsymbol{R}_{\hat{z}}(\beta_t)$ um Winkel β_t rotierten Vertexkoordinaten $\hat{\boldsymbol{x}}\big|_t$ und den undeformierten Koordinaten $\hat{\boldsymbol{x}}\big|_{t_0}$ basierend auf den Vertices v_i des Evaluationsgraphen G_{E} berechnet. Die Extrusionsrichtung, um die der deformierte Querschnitt rotiert wird, wird mit $\hat{z}$ angegeben. Zuletzt wird daraus das arithmetische Mittel über alle $|\mathcal{V}^{\{G_{\mathrm{E}}\}}|$ Vertices des Evaluationsgraphen gebildet. Das zu lösende Optimierungsproblem lautet für jeden Zeitpunkt t

$$\begin{aligned} &\min_{\beta_t} && \frac{1}{|\mathcal{V}^{\{G_{\mathrm{E}}\}}|} \sum_{v_i \in \mathcal{V}^{\{G_{\mathrm{E}}\}}} \left\| \boldsymbol{R}_{\hat{z}}(\beta_t)\hat{\boldsymbol{x}}\big|_t - \hat{\boldsymbol{x}}\big|_{t_0} \right\|_2, \\ &\text{so dass} && -180^\circ \leq \beta_t \leq 180^\circ. \end{aligned}$$

In Abschnitt 5.5.5 wurde bereits angesprochen, dass Zellen eine Mindestdeformation aufweisen müssen, um im Reset-Modul überhaupt als möglicher Kandidat zur Optimierung gewählt zu werden. Mit der Einführung des Formabweichungsmaßes $\tilde{A}_\Delta$ kann der zugehörige Grenzwert auch quantifiziert werden. Eine Zelle muss einen Formabweichungswert zwischen $0{,}05 \leq \tilde{A}_\Delta \leq 1{,}0$ vorweisen, um als ausreichend deformiert zu gelten. Ebenso entscheidet der Formabweichungswert auch darüber, ob die Umgebung die Episode terminiert. Das geschieht dann, wenn $\tilde{A}_\Delta \leq 0{,}05$ ist.

Auf Basis des Formabweichungswerts kann ebenfalls geklärt werden, wie im Reset-Modul die kritischste Zelle gewählt wird. Wie wichtig es ist, eine konkrete Zelle zu optimieren, folgt daraus, wie stark sie deformiert ist und wie groß die Fläche ist, die sie aufspannt. Zwar ist die Zellengröße für die Optimierung des Agenten nicht direkt entscheidend, doch bei der Anwendung des Agenten als Heuristik, soll möglichst eine Zelle gewählt werden, die besonders einflussreich auf die Zielfunktion der Optimierungsaufgabe ist. Je größer die von der Zelle aufgespannte Fläche ist, desto mehr Einfluss hat diese Zelle im Mittel auf das gesamte Strukturverhalten. Für jede im Graphen vorhandene Zelle k wird der geschätzte Einfluss auf das Strukturverhalten κ über

$$\kappa_k := \tilde{A}_{\Delta,k} \cdot A_k\Big|_{t_0} \tag{5–6}$$

berechnet. Zur Optimierung wird die Zelle mit dem maximalen geschätzten Einfluss auf das Strukturverhalten herangezogen.

5.5.8 Reward-Funktion

Zur Beurteilung, ob eine vom Agenten gewählte Kante die Zelle sinnvoll versteift, wird die relative Verbesserung $\delta_{t,\mathrm{rel}}$ der Formabweichung der Zelle von der Zelle ohne und der Zelle mit der neu eingebrachten Kante berechnet. Die Betrachtung der relativen Verbesserung ist wichtig, da das mechanische Verhalten verschiedener Zellen höchst unterschiedlich ist und somit die Strukturverbesserung auf Basis des vorhandenen Verbesserungspotentials bewertet wird, anstelle einer absoluten Bewertung. Dadurch wird das Training des Agenten vereinfacht, da sich die Rewards stets in der gleichen Größenordnung befinden. Die relative Verbesserung bzgl. der Steifigkeit der Zelle ergibt sich zu dem aktuellen Zeitpunkt t aus den Formabweichungswerten $\tilde{A}_\Delta$ über

$$\delta_{t,\mathrm{rel}} = \mathrm{clip}\left(\frac{\tilde{A}_\Delta(s_{t-1}) - \tilde{A}_\Delta(s_t)}{\tilde{A}_\Delta(s_0)}; -1; 1\right). \tag{5–7}$$

Das Clipping der relativen Verbesserung in den Wertebereich $[-1; 1]$ erfüllt zwei Aufgaben.

1. Der Agent wird implizit bestraft, wenn dieser als erstes eine Kante in die Zelle einbringt und das daraus resultierende Formabweichungsmaß schlechter ist als das der initialen, leeren Zelle und dann eine besonders gute Kante einbringt. In genau solchen Fällen würde die relative Verbesserung einen Wert > 1 erzeugen, welcher durch das Clipping begrenzt ist. Vorhersagen, wo zuerst eine schlechte und dann eine besonders geeignete Kante eingebracht wird, sind risikobehaftet, da der Agent den nächsten Zustand abschätzen muss, um die zunächst ungünstig weichere Zelle signifikant zu versteifen. Durch das Clipping wird der Agent dazu ermutigt, zuerst die Kante mit niedrigerem Risiko einzubringen.

2. Das Clipping verhindert eine Gefährdung des Trainings durch Werte außerhalb des üblichen Wertebereiches, welche durch potentiell auftretende numerische Problemfälle entstehen könnten. Es handelt sich um eine rein prophylaktische Vorsichtsmaßnahme.

Der Reward r_{t+1} bestimmt sich final aus dem Ausdruck

$$r_{t+1} = p_{\text{ins}} + \begin{cases} \delta_{t,\text{rel}} & \text{falls das Modell fertigbar ist,} \\ -1 & \text{sonst.} \end{cases} \tag{5–8}$$

Der Bestrafungsterm p_{ins} dient dazu, dass der Agent nur Kanten in die Zelle einbringen darf, die das Steifigkeitsverhalten der Zelle auch signifikant verbessern. Dadurch sollen unnötig komplexe Zellen vermieden werden und dient als Motivation für den Agenten sinnvolle Zellen mit möglichst wenig eingezogenen Kanten zu finden, ohne dabei die Performance der optimierten Zelle übermäßig stark negativ zu beeinflussen. Festgelegt wird der Bestrafungsterm mit einem Wert von $p_{\text{ins}} = -0{,}1$. Um einen positiven Reward r_{t+1} zu generieren, muss die Struktur fertigbar sein und eine relative Verbesserung der Zellsteifigkeit $\delta_{t,\text{rel}} > 0{,}1$ aufweisen.

Es sei nochmal darauf hingewiesen, dass der Reward nur dann berechnet werden kann, wenn das Modell herstellbar ist und die Simulation des Modells in Konsequenz durchgeführt wurde. Wenn das Modell nicht herstellbar ist, wird der Agent für die eingebrachte Kante bestraft und die Episode terminiert, da die Zelle durch das Einbringen weiterer Kanten nicht verbessert bzw. wieder herstellbar werden kann.

5.5.9 Aktionsraum

Der Aktionsraum definiert die Art und Weise der vom Agenten ausgewählte Aktion und wie diese dann in eine Topologieänderung der Zelle übersetzt wird. Bevor der Aktionsraum der hier gezeigten Umgebung genauer beschrieben werden kann, muss zunächst auf die verfügbaren Raumtypen eingegangen werden. Die Raumtypen werden von GYM zur Verfügung gestellt und von STABLE-BASELINES3 genutzt. Diese Raumtypen unterscheiden sich nach Art und Struktur der betrachteten Daten. Alle genannten Räume lassen sich sowohl für Aktionen, als auch für Zustände bzw. Observationen nutzen und repräsentieren somit den Aktionsraum $\mathcal{A}$ bzw. den Zustands-/Observationsraum $\mathcal{S}$ aus der Definition eines MEPs in Abschnitt 3.4.2.

Die folgenden Räume stehen in STABLE-BASELINES3 zur Verfügung:

- Der *diskrete* Raum wird durch eine Liste an Ganzzahlen repräsentiert.

- Der *multidiskrete* Raum wird durch das kartesische Produkt aus diskreten Räumen aufgebaut. Für ein Beispiel mit zwei diskreten Räumen lässt sich der zugehörige multidiskrete Raum als Tabelle vorstellen, bei der jeder Tabelleneintrag eine einzigartigen Kombination der Ganzzahlen aus den jeweiligen diskreten Räumen darstellt.
- Der *multibinäre* Raum besteht aus einer Liste aus mehreren binären Zuständen.
- Der *Box*-Raum wird durch das kartesische Produkt aus von dem/der Nutzer*in spezifizierten Intervallen gebildet und kann somit kontinuierliche Daten verarbeiten. Falls zwei Intervalle von dem/der Nutzer*in vorgegeben werden bedeutet das, dass der zugehörige Box-Raum jede Kombination aus Werten der beiden Intervalle abbilden kann.
- Der *Dict*-Raum fasst verschiedene der o. g. Räume je Feature zusammen und ordnet dabei jedem Feature einen Subraum zu.

Die Bibliothek GYM stellt noch weitere Räume zur Verfügung, welche nicht in STABLE-BASELINES3 umgesetzt sind. Nennenswert ist hier der *Graph*-Raum, welcher sich potentiell für die vorliegende, graphbasierte Umgebung eignen würde.

Auf Basis der vorliegenden Raumtypen kann nun die Struktur des Aktionsraums bestimmt werden. Der Aktionsraum repräsentiert die Menge an Aktionen, die der Agent in einer Umgebung ausführen kann. Der Agent soll zum einen eine Kanten benennen können, die in die Zelle eingebracht wird. Die möglichen einziehbaren Kanten der Zelle lassen sich durch eine ganzzahlige ID ansteuern, sodass ein kontinuierlicher Raum nicht nötig ist. Ebenso soll der Agent steuern können, ob die Episode terminieren soll, oder nicht. Auch hier eignet sich eine Ganzzahl zur Beschreibung. Folglich wird für die Umgebung der multidiskrete Raum genutzt, bei dem verschiedene Teilaktionen durch die jeweilige Angabe einer Ganzzahl zu einer Aktion zusammengefasst werden.

5.5.10 Durchführung der Topologieänderungen

Der Agent benennt durch ein Element aus dem zuvor definierten Aktionsraum eine Kante auf Basis einer Kanten-ID zwischen 0 und $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}| - 1$. Diese ID beschreibt die Position der zu aktivierenden Kante im Adjazenzvektor. Jedem Eintrag im Adjazenzvektor ist, unabhängig von der exakten Form Zelle, eine Vertexkombination von Vertices auf dem Rahmen der Zelle zugeordnet. So kann der Agent konsistent ansteuern, welche Kante in der Zelle eingezogen werden soll. Wenn die gleiche Kante mehrfach aktiviert wird, ändert sich die Topologie und das mechanische Verhalten der Zelle nicht.

Für den Fall, dass mindestens eine Kante bereits in die Zelle eingebracht wurde, besteht die Möglichkeit, dass sich eine weitere Kante mit der eingebrachten Kante im modifizierten Zellengraph G_{Zm} schneidet. Zur Auflösung des Schnitts bei der Bildung des Zellengraphen G_{Z} und des Strukturgraphen G_{S}, wird im Schnittpunkt ein neuer Vertex eingebracht.

Die beiden sich schneidenden Kanten werden im Schnittpunkt aufgetrennt und die vier entstandenen Kanten mit dem neuen Vertex verbunden. Für den Fall, dass eine dritte Kante eingebracht wird, kann es passieren, dass diese durch einen bereits existierenden Schnittpunkt verläuft. Dann wird dieser Schnittpunkt verwendet.

5.5.11 Observationsraum

Das mechanische Verhalten einer Zelle kann durch verschiedenste Größen und Kennwerte, den Features, beschrieben werden. Diese Features bilden alle relevanten Größen für den Agenten zur Entscheidungsfindung. Um möglichst viele verschiedenartige Teilobservationen in einer Observation zusammenzufassen, eignet sich der Dict-Raum, welcher jedem Feature einen Subraum zuordnet. Ein sauberes und durchdachtes Feature-Engineering ist wichtig, um den Lernprozess des Agenten so stark wie möglich zu vereinfachen und die Performance des trainierten Agenten somit zu verbessern.

Wichtig ist anzumerken, dass im final trainierten Agenten nicht zwingend alle hier vorgestellten Features genutzt werden. Der Auswahlprozess zur Findung von Features des finalen Observationsraums wird später gezeigt.

Datenaufbereitung: Bevor auf die detaillierte Beschreibung der Features eingegangen wird, werden einige grundlegenden Besonderheiten der Beschreibung der Features im Observationsraum geklärt. Zentrale Größen, die das Feature und den zugehörigen Subraum beschreiben sind die Dimension und Form eines Features. Die Dimension eines Features gibt an, durch wie viele Werte das Feature beschrieben wird. Je geringer die Dimension des Features ist, desto weniger Werte muss der Agent anlernen, was die Sample-Effizienz und Konvergenz des Agenten verbessert. Die Form gibt an, wie die Werte tabellarisch strukturiert sind. Beide Größen sind abhängig von der Anzahl der Seiten einer Zelle $|\mathcal{S}^{\{G_{\mathrm{Zm}}\}}|$ und der Anzahl an Vertices auf dem Rahmen der Zelle $|\mathcal{V}^{\{G_{\mathrm{Zm}}\}}|$.

Einige Features werden basierend auf geometrischer Kenngrößen direkt in der Umgebung normiert. Dieses Vorgehen wird *Geometrische Normierung* (GN) genannt, da sich die Normierungswerte aus geometrischen Kennwerten ableiten lassen. Ziel ist es, die den Features zugehörigen Werte zwischen verschiedene Zellen vergleichbar zu machen, in dem Abhängigkeiten zu geometrischen Größen herausgerechnet werden. Das konkrete Vorgehen hängt vom betrachteten Feature ab und wird in den folgenden Beschreibungen der Features aufgegriffen.

Wenn in einer Zelle bereits eine Kante eingezogen wurde und der Agent eine zweite Kante einbringt, dann ist es möglich, dass sich die Kanten schneiden. Das ist problematisch, da die Anzahl der Schnittpunkte und damit die Anzahl der Vertices und Kanten innerhalb der

Zelle bei unterschiedlich vielen eingezogenen Kanten variabel ist. Die zwingend geforderte konsistente Beschreibung der Zelle und der abgeleiteten Observationen erfordert allerdings, dass die Anzahl der Vertices und Kanten innerhalb der Zelle konstant ist.

Deshalb werden die Strukturantworten aller gleich orientierten und geteilten Kanten zwischen der direkten Verbindungen zweier Vertices auf dem Zellrahmen zusammengefasst. Dieses Vorgehen nennt sich *Aggregation*. Darunter versteht man das Zusammenfassen von Werten, die dann durch einen einzelnen Skalar repräsentiert werden. Entlang dieser Kanten eines Kantensegments wird eine Aggregationsfunktion angewandt. Diese Funktion fasst die Strukturantworten zu einer einzelnen, für das gesamte Segment repräsentativen Strukturantwort, zusammenzufassen. Dadurch kann die konsistente Beschreibung der Observationen aufrecht erhalten bleiben und die Strukturantworten können dem Adjazenzvektor (vgl. Abbildung 5–8) zugeordnet werden. Abbildung 5–15 visualisiert das Konzept der Aggregation der Strukturantworten.

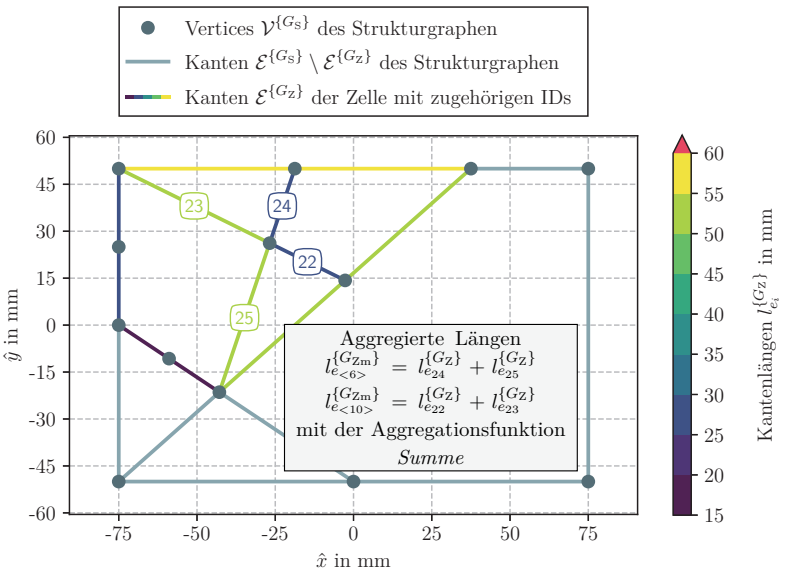


Abbildung 5–15: Aggregation von Strukturantworten am Beispiel von Kantenlängen $l_{e_i}^{\{G_Z\}}$. Die aggregierten Längen sind $l_{e_{<6>}}^{\{G_{Zm}\}}$ und $l_{e_{<10>}}^{\{G_{Zm}\}}$, wobei die Notation von <6> und <10> den Index der zugehörigen Kante im Adjazenzvektor angibt

Zur konsistenten Beschreibung der Zellen werden kantenbezogene Features, wie die Kantenlänge und die innere Energiedichte je Kante, stets für alle möglichen Kanten der Zelle beschrieben. Das ist unabhängig davon, ob diese bereits vom Agenten eingezogen wurden oder diese noch eingezogen werden können und kein aktiver Bestandteil der

Struktur sind. Wenn Kanten noch nicht eingezogen worden sind, können die zugehörigen Einträge natürlich nicht mit Simulationsdaten oder geometrischen Daten bestückt werden. Entsprechend werden die Featurevektoren mit Dummy-Werten gefüllt. Dieses Vorgehen wird als *Imputation* bezeichnet. Die Imputation der Featurevektoren gilt auch für den Fall, dass das Modell nicht herstellbar ist, weil es die Herstellrestriktionen verletzt. Dann liegen keine Simulationsdaten vor. Der Agent erwartet aber trotzdem eine Observation. Auch in diesem Fall wird die Observation mit Dummy-Werten gefüllt. Für alle Features, die imputiert werden müssen, ist der Imputationswert 0. Eine Ausnahme ist das Deformationsbild, bei dem der Wert 255 genutzt wird, welcher jedem Farbkanal in jedem Pixel zugewiesen wird. Das resultiert in einem komplett weißen Bild, wenn keine Simulationsdaten vorhanden sind. Dieser Wert wird gewählt, da der Hintergrund der Bilder so stets weiß ist, unabhängig davon, ob die tatsächliche Deformation visualisiert wird oder nicht.

Koordinaten: Die Koordinaten beschreiben die Größe und Form einer Zelle, welche Einfluss auf die Herstellbarkeit und das mechanische Verhalten der Struktur haben. Visualisiert wird das Koordinaten-Feature an einem Beispiel in Abbildung 5–16.

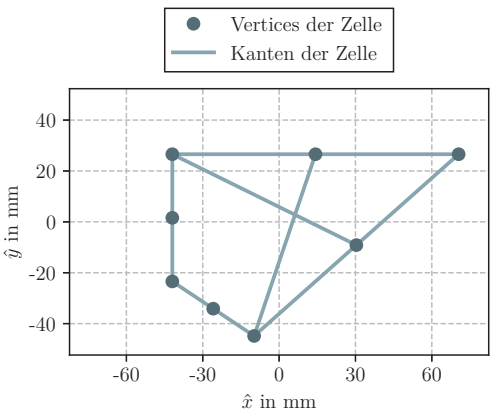


Abbildung 5–16: Positionierung der Zellvertices $\mathcal{V}^{\{G_{\mathrm{Zm}}\}}$, deren Koordinaten das Koordinaten-Feature beschreiben

Gemessen werden die Koordinaten an den Vertices auf dem Rahmen der Zelle entlang der lokalen $\hat{x}$- und $\hat{y}$-Achse im Profilkoordinatensystem, d. h., dass die Koordinaten auf den Schwerpunkt der Zelle bezogen sind. Die Einträge im Featurevektor sind entlang des Rahmens der Zelle sortiert und lassen sich durch Werte der Form $|\mathcal{V}^{\{G_{\mathrm{Zm}}\}}| \times 2$ beschreiben. Der Faktor 2 ergibt sich daraus, dass jeder der Vertices durch zwei Raumrichtungen beschrieben wird. Dadurch, dass die Koordinaten eine kontinuierliche Größe sind, werden

diese durch einen Box-Subraum tabellarisch beschrieben. Die Zeilen des tabellarischen Box-Subraums entsprechen den $|\mathcal{V}^{\{G_{\mathrm{Zm}}\}}|$ Vertices entlang des Zellrahmens und die Spalten entsprechen der betrachteten Richtung. Sortiert sind die Vertices entsprechend der bereits eingeführten Konvention gegen den Uhrzeigersinn entlang des Zellrahmens.

Da die Größe der Zelle in den Koordinaten vorhanden bleiben soll, werden diese nicht geometrisch normiert. Bestimmt werden die Koordinaten ausschließlich zum Zeitpunkt t_0, da die Koordinaten die initiale, undeformierte Form der Zelle beschreiben sollen. Dadurch, dass Vertices und keine sich potentiell schneidenden Kanten beschrieben werden, ist keine Aggregation von Werten nötig. Ebenso ist keine Imputation nötig, da die geometrischen Werte simulationsunabhängig zur Verfügung stehen.

Verschiebungen: Die Verschiebungen sind analog aufgebaut wie die Koordinaten und haben das Ziel die Deformation der Zelle für den Agenten zu beschreiben. Sie werden ebenfalls im lokalen Profilkoordinatensystem für alle Vertices entlang des Rahmens ermittelt, wodurch der Featurevektor durch Werte der Form $|\mathcal{V}^{\{G_{\mathrm{Zm}}\}}| \times 2$ repräsentiert wird. Als Subraum eignet sich der tabellarisch aufgebaute Box-Raum. Das Feature der Verschiebungen wird in Abbildung 5–17 dargestellt.

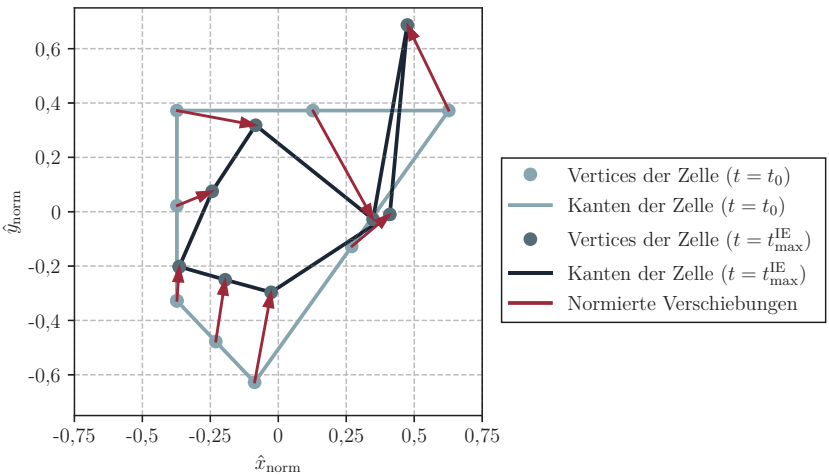


Abbildung 5–17: Visualisierung der normierten Verschiebungen des Features. Für die Darstellung dieses Features wird eine leere Zelle ohne eingezogene Kanten betrachtet

Ermittelt werden die Verschiebungen an der Position der Vertices $\mathcal{V}^{\{G_{\mathrm{Zm}}\}}$ des modifizierten Zellengraphen. Dazu werden die am nächsten gelegenen Vertices der Evaluationsgraphen $G_{\mathrm{E}}(t_0)$ und $G_{\mathrm{E}}(t^{\mathrm{IE}}_{\mathrm{max}})$ ermittelt und deren Verschiebung den Vertices $\mathcal{V}^{\{G_{\mathrm{Zm}}\}}$ zugewiesen.

Damit die Verschiebungen unabhängig von der Größe der Zelle sind, wird eine geometrische Normierung über die Zellengröße in der Trainingsumgebung vorgenommen. Zur geometrischen Normierung wird ein Rechteck mit den Maßen $\Delta\hat{x}$ und $\Delta\hat{y}$ konstruiert, welches die Zelle mit minimaler Fläche umschließt. Die einzelnen Verschiebungen $u_{\hat{x}}$ werden dann mit $\Delta\hat{x}$ normiert. Analog wird für die Verschiebungen in $\hat{y}$-Richtung vorgegangen.

Der Auswertezeitpunkt der Verschiebungen ist $t_{\max}^{\mathrm{IE}}$. Das ist der Zeitpunkt, an dem die innere Energie der Zelle über die Zeit maximal ist. Dadurch soll der Agent darauf trainiert werden, die Steifigkeit der Zelle für den meist deformierten Zustand zu erhöhen.

Kantenlängen: Die Kantenlängen werden für jede Kante der Zelle ermittelt und sollen dem Agenten beispielsweise Auskunft über die Knickneigung einer Kante geben, da die kritische Knicklast signifikant von der Knicklänge abhängt. Diese Information soll dazu dienen, dass der Agent abschätzen kann, welche Wirkung das Einbringen einer Kante auf die Zelle hat und ob die Gefahr des Knickens eines anderen Strukturelements besteht. Theoretisch könnte sich der Agent die Kantenlängen aus den Koordinaten selbstständig ermitteln. Um dem Agenten die nicht nötige und anzulernende Informationsextraktion abzunehmen, werden diese zusätzlich in Form eines Box-Subraums im Observationsraum verwendet. Abbildung 5–18 stellt den Featurevektor vor. Die Einträge des Vektors sind den im Beispiel eingezogenen Zellkanten zugeordnet.

Die Anzahl der beschreibenden Werte bzw. die Länge des Featurevektors ergibt sich aus der Anzahl an Vertices entlang des Rahmens $|\mathcal{V}^{\{G_{\mathrm{Zm}}\}}|$ und der Anzahl an vom Agenten in die Zelle einziehbaren Kanten $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|$. Das umfasst alle Kanten der Zelle $|\mathcal{E}_{\mathrm{ges}}^{\{G_{\mathrm{Zm}}\}}|$. Zur Beschreibung in einem Vektor werden die sortierten Kantenlängen entlang des Zellrahmens und die theoretisch einziehbaren Kanten innerhalb der Zelle konkateniert. Die Sortierung folgt der Ordnung der Vertices entlang des Rahmens und der Sortierung des Adjazenzvektors.

Ausgewertet werden die Kantenlängen auf dem Strukturgraphen zum Zeitpunkt t_0. Die Aggregation von Kantenlängen lässt sich leicht umsetzen, denn die Länge des Kantensegments lässt sich als Summe der Einzelkanten auffassen. Dabei ändert sich die mechanische Bedeutung der Kantenlänge nicht. Die Imputation mit dem Wert 0 mm ist ebenfalls aus mechanischer Sicht sinnvoll, da ein solcher Wert bedeutet, dass die entsprechende Kante nicht vorhanden bzw. eingezogen ist. Für den Fall, dass die Simulation nicht durchgeführt werden kann, weil die Struktur nicht herstellbar ist, wird keine Imputation benötigt, da die Kantenlängen im Strukturgraphen stets vorhanden sind.

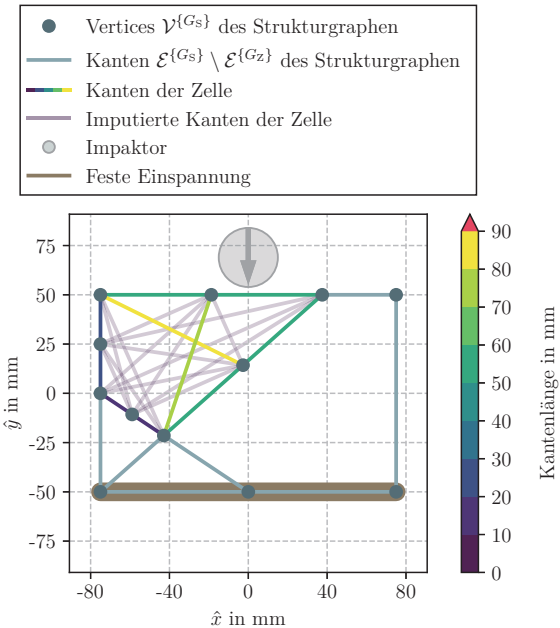


Abbildung 5–18: Visualisierung des Kantenlängen-Features anhand einer Beispielzelle mit zwei bereits eingezogenen Kanten

Wandstärke: Die Wandstärke ist ein integraler Bestandteil in der Beschreibung der Zellsteifigkeit. Je dicker die Wand ist, desto mehr Widerstand leistet diese gegen Deformation. Ebenso ist die Knickneigung von Wänden abhängig von der Wandstärke. Je dünner die Wand ist, desto geringer ist die kritische Knicklast, was in einem früheren Knicken der Wand resultiert. Neben den mechanischen Eigenschaften dient die Wandstärke auch dazu, dass der Agent abschätzen kann, wie weit er von den Grenzen der zulässigen Wandstärken entfernt ist. Das soll verhindern, dass der Agent eine weitere Kante einzieht und durch die Massenrestriktion die Wandstärke unterhalb des zulässigen Bereichs fällt.

Es wird lediglich ein einzelner Wert in einem Box-Raum benötigt, um die Wandstärke für die Observation zu beschreiben, da diese für alle Wände der Struktur gleich ist. Dadurch, dass der Wert im Strukturgraphen stets verfügbar ist, muss kein Wert für die Imputation definiert werden.

Gesamtlänge: Die Gesamtlänge ist das einzige Feature, welches nicht nur die Zelle, sondern die Kantenlänge des gesamten Graphen beschreibt. Es berechnet sich aus der Summe aller Kantenlängen im Strukturgraphen. Das Ziel dieses Features ist, dass der

Agent einschätzen kann, wie stark die Änderung der Wandstärke durch die Massenrestriktion sein wird, wenn eine neue Kante in den Graphen durch den Agenten eingebracht wird. So kann der Agent erkennen, ob die Wandstärke durch eine neue Kante zu klein wird und das Modell somit nicht mehr fertigbar wäre. Dadurch, dass die Wandstärke im gesamten Graphen gleich ist und die Kantenlänge zur Masse der Struktur proportional ist, kann der Agent theoretisch die Änderung der Wandstärke durch das Einbringen einer neuen Kante bestimmen.

Es handelt sich bei der Gesamtkantenlänge um einen Skalar, welcher durch einen Box-Raum beschrieben wird. Eine geometrische Normierung ist nicht nötig. Ermittelt wird die Gesamtkantenlänge im undeformierten Zustand. Da der Wert stets berechnet werden kann, muss kein Wert für die Imputation festgelegt werden.

Herstellbarkeit: Auf Basis fünf verschiedener Kriterien wird überprüft, ob der Graph herstellbar ist. Dadurch kann der Agent detektieren, zu welcher Verletzung der Herstellungsrestriktion die gewählte Aktion geführt hat. Außerdem ist so leicht erkennbar, wenn die Observation aus hauptsächlich imputierten Größen besteht, weil keine Simulation durchgeführt wurde.

Da sich die Erfüllung der einzelnen Kriterien durch einen binären Wert repräsentieren lässt, wird das Feature durch einen multibinären Raum repräsentiert. Getestet wird,

- ob der Graph nicht-planar ist, d. h., dass sich Kanten ohne Vertices im Schnittpunkt durchdringen,
- ob mindestens ein Abstand zwischen zwei Kanten die Herstellungsrestriktion verletzt,
- ob ein Winkel zwischen benachbarten Kanten existiert, der die Herstellungsrestriktion verletzt,
- ob mindestens eine Kante kürzer ist, als durch die Herstellungsrestriktion vorgegeben wird und
- ob die Wandstärke den zulässigen Bereich verlässt.

Trifft mindestens eines dieser Kriterien zu, ist der Graph nicht herstellbar. Die exakten Herstellungsrestriktionen wurden bereits in Tabelle 5–3 aufgezeigt. Da es sich um multibinäre Werte handelt, müssen diese nicht normiert werden und aufgrund der simulationsunabhängigen Verfügbarkeit der Werte auch nicht imputiert werden.

Zellenanbindung: Es ist sinnvoll, dass der Agent dort Kanten einzieht, wo er sich zusätzliche Steifigkeit aus der Struktur außerhalb der Zelle holen kann. Das ist gerade an den Vertices auf dem Zellrahmen möglich, die mit der äußeren Struktur verbunden sind. Ob ein Vertex mit der äußeren Struktur verbunden ist, wird über den Vertexgrad des betrachteten Rahmenvertex ermittelt. Die Dimension des Features entspricht der Anzahl an Rahmenvertices $|\mathcal{V}^{\{G_{\mathrm{Zm}}\}}|$, an denen die Zellenanbindung überprüft wird. Da es sich auch hier, analog zu der Herstellbarkeit, um binäre Kriterien handelt, wird das Feature durch einen multibinären Raum beschrieben. Die Definition eines Wertes zur Imputation ist nicht nötig, da alle benötigten Informationen aus dem Strukturgraphen simulationsunabhängig extrahiert werden können. Abbildung 5–19 zeigt die Zellenanbindung an die äußere Struktur exemplarisch.

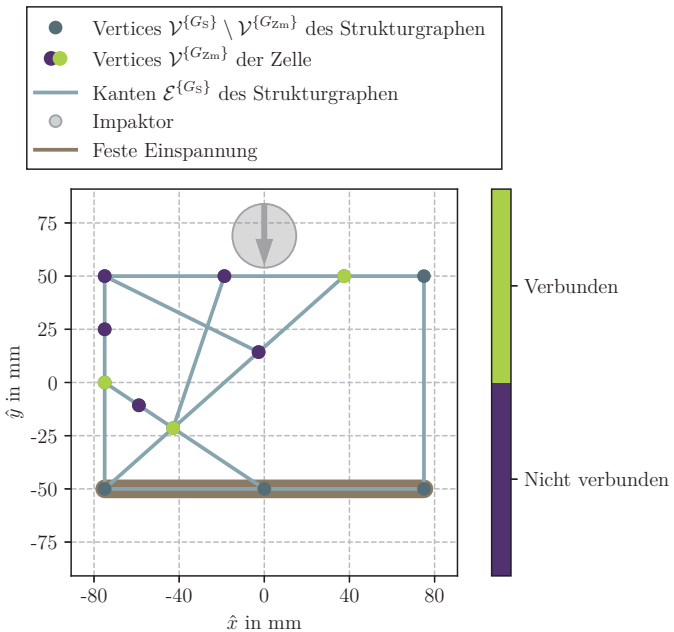


Abbildung 5–19: Visualisierung des Zellenanbindungs-Features anhand einer Beispielzelle mit zwei bereits eingezogenen Kanten

Adjazenzvektor: Der Adjazenzvektor beschreibt, welche durch den Agenten theoretisch einziehbaren Kanten in der Zelle miteinander verbunden sind und wurde bereits in Abbildung 5–8 visualisiert. Das soll dem Agenten helfen zu identifizieren, welche Kanten im aktuellen Zustand eingezogen sind und welche Kantenwerte anderer Features imputiert

sind. Die Dimension des Adjazenzvektors ist $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|$ und folgt aus Gleichung (5–1). Ob eine Kante eingezogen ist oder nicht, lässt sich durch einen multibinären Raum beschreiben. Es ist keine Normierung und keine Imputation nötig.

Innere Energiedichte: Die innere Energie einer Wand gibt an, wie viel der vorhandenen Energie im Crashprozess von ihr absorbiert wurde und kann als Deformationsenergie verstanden werden. Bestandteile der inneren Energie sind die elastische Verzerrungsenergie und die bei der plastischen Deformation geleistete Arbeit. Der Agent hat mit diesen Informationen die Möglichkeit, Bereiche der Zelle zu identifizieren, welche besonders durch eine modifizierte Topologie unterstützt werden müssen, um Steifigkeit für die Zelle zu generieren. Die innere Energie wird aufgrund der kontinuierlichen Natur des Features durch einen Box-Raum beschrieben.

Der Auswertezeitpunkt der inneren Energie der Zellwände ist $t^{\mathrm{IE}}_{\max}$. So wird die Situation an dem Zeitpunkt beschrieben, wo die Strukturen die meiste Deformationsenergie absorbiert haben und für den die Steifigkeit der Zelle ausgelegt werden muss.

Zur Vergleichbarkeit von verschiedenen Zellen müssen die inneren Energien normiert werden. Eine in der Crashentwicklung bewährte Herangehensweise ist, dass die innere Energie $E_{\mathrm{i},k}$ mit dem zugehörigen Volumen V_k der betrachteten Strukturkomponente k durch

$$\tilde{E}_{\mathrm{i},k} = \frac{E_{\mathrm{i},k}}{V_k} \tag{5–9}$$

normiert wird. Zur Übersichtlichkeit werden die Zeitpunkte zur Ermittlung der inneren Energie nicht notiert. Man erhält ein Maß dafür, wie viel Energie von der Struktur je Einheitsvolumen absorbiert wird. Das Volumen V_k wird in der Ausgangskonfiguration t_0 bestimmt. Das Maß nennt sich innere Energiedichte und ist unabhängig von der konkreten Form und Größe der Zelle. Deshalb eignet sich das Maß zur Vergleichbarkeit zwischen verschiedenen Zellen und folglich auch zur Entscheidungsfindung für den Agenten.

Bei der Aggregation der inneren Energiedichten muss darauf geachtet werden, dass aufgrund der Normierung mit dem Volumen diese nicht einfach aufaddiert werden dürfen und dann als repräsentativer Wert für das Kantensegment herangezogen werden dürfen, schließlich ist i. Allg.

$$\sum_k \tilde{E}_{\mathrm{i},k} \neq \frac{\sum_k E_{\mathrm{i},k}}{\sum_k V_k}. \tag{5–10}$$

Stattdessen wird ein mit der Kantenlänge gewichteter Mittelwert der Energiedichten verwendet, um auf eine korrekte repräsentative Energiedichte für das gesamte Kantensegment zu schließen. Aus dem gewichteten Mittel der Energiedichten lässt sich

$$\frac{\sum_k l_{e_k}^{\{G_{\mathrm{Z}}\}} \tilde{E}_{\mathrm{i},k}}{\sum_k l_{e_k}^{\{G_{\mathrm{Z}}\}}} = \frac{\sum_k l_{e_k}^{\{G_{\mathrm{Z}}\}} \frac{E_{\mathrm{i},k}}{V_{e_k}^{\{G_{\mathrm{Z}}\}}}}{\sum_k l_{e_k}^{\{G_{\mathrm{Z}}\}}} = \frac{\sum_k \frac{E_{\mathrm{i},k}}{A_{e_k}^{\{G_{\mathrm{Z}}\}}}}{\sum_k l_{e_k}^{\{G_{\mathrm{Z}}\}}} = \frac{\sum_k E_{\mathrm{i},k}}{\sum_k V_{e_k}^{\{G_{\mathrm{Z}}\}}} \tag{5–11}$$

mit der Querschnittsfläche $A_{e_k}^{\{G_Z\}} = \text{const.}\ \forall e_k \in \mathcal{E}^{\{G_Z\}}$ ableiten. Die Querschnittsfläche $A_{e_k}^{\{G_Z\}}$ ist in einem Entwurf deshalb konstant, weil sie von der Extrusionstiefe des Modells und der Wandstärke des Modells abhängig ist, wobei beide Parameter per Definition konstant sind.

Abbildung 5–20 zeigt eine Darstellung der inneren Energiedichte für die Rahmenkanten und die vom Agenten bereits eingezogenen Zellkanten. Alle nicht eingebrachten Kanten werden mit dem Imputationswert von $0\,\mathrm{mJ/mm^3}$ repräsentiert.

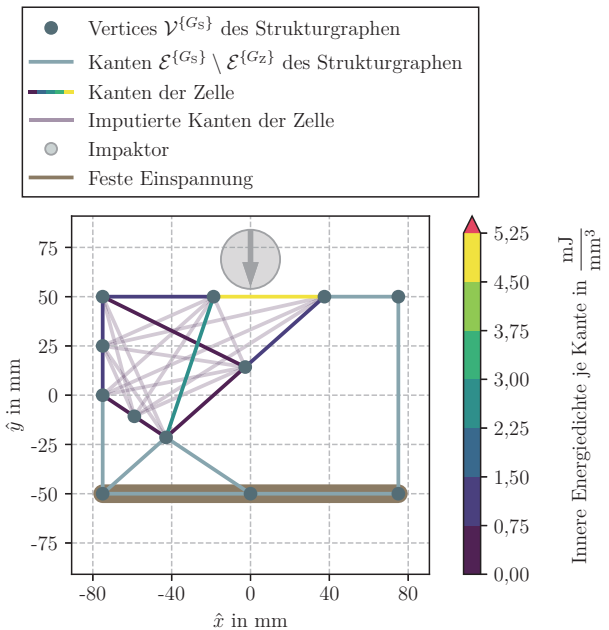


Abbildung 5–20: Visualisierung des Energie-Features anhand einer Beispielzelle mit zwei bereits eingezogenen Kanten

Spannungen: Zur Berechnung der Spannungen innerhalb eines Querschnitts einer Wand, wird zunächst geklärt, wie die zugehörigen Schnittkräfte bestimmt werden, aus denen die Spannungen abgeleitet werden. Unter Schnittkräften werden in der Mechanik innere Kräfte verstanden, welche entlang einer Schnittfläche wirken, um äußere Belastungen auszugleichen. Schnittkräfte sind eine vektorielle Größe und haben daher eine Richtung und eine Länge. Die Schnittkräfte sollen individuell für jede Kante im Graphen im Featurevektor angegeben werden.

Zum Auslesen der Schnittkräfte muss automatisiert eine Schnittfläche für jede Kante der Zelle in das Simulationsmodell eingebracht werden. Der Normalenvektor der Fläche zeigt dabei in die Richtung des Richtungsvektors der betrachteten Kante. Die Schnittflächen werden, wie in Abbildung 5–21 dargestellt wird, mittig von jeder Kante positioniert und erstrecken sich im Trainingsmodell über die gesamte Extrusionstiefe.

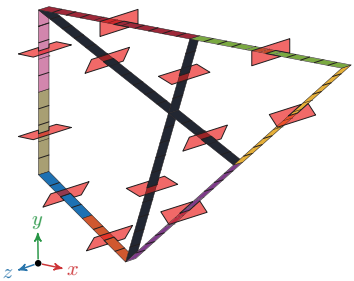


Abbildung 5–21: Positionierung der Schnittflächen zur Schnittkraftberechnung

Um eine konsistente mechanische Beschreibung von Schnittkräften zwischen verschiedenen Wänden zu ermöglichen, wird die Richtung der Schnittkräfte in einem lokalen Koordinatensystem ermittelt. Das lokale Koordinatensystem bezieht sich auf die Orientierung der betrachteten Kante. Dadurch wird die Angabe der Schnittkraft als Axial- und als Lateralkraft ermöglicht. Die Axialkraft wirkt in der Querschnittsebene entlang der Kante in Form einer Zug- bzw. Druckkraft und die Lateralkraft wirkt orthogonal in Form einer Querkraft. Da sich die aus den Kanten abgeleiteten Wände über die Zeit deformieren und sich damit auch der Normalenvektor der Wand an der Auswertestelle ändert, wird das lokale Koordinatensystem so definiert, dass es sich mit dem Normalenvektor am Mittelpunkt der Wand mitdreht. Schnittkräfte in Extrusionsrichtung werden aufgrund der geringen Extrusionstiefe des Trainingsmodells nicht betrachtet. Entsprechend können auch keine Effekte durch Kräfte in diese Richtung quantifiziert werden. Die Schnittfläche wird dort aufgespannt, wo das Simulationsmodell zum Aufbau des Evaluationsgraphen ausgelesen wird. Das ist i. d. R. die Mitte des Extrusionsprofils entlang der Extrusionsachse.

Durch das Vorzeichen der Axialkräfte soll der Agent detektieren, ob Zug- oder Drucklasten in der Wand wirken und somit ggfs. eine Knickgefahr für die Wand besteht. Durch den Betrag der Axialkraft soll der Agent abschätzen können, wie kritisch die Knicklasten sind. Lateralkräfte charakterisieren die strukturelle Integrität der Zelle weiter. Bereits sehr kleine Lateralkräfte tragen dazu bei, dass die Strukturkomponente durch die Instabilität bei einer Knickbelastung kollabiert und das Knicken begünstigt wird. Durch die Informationen über die aktuelle Lastverteilung in der Zelle, kann der Agent idealerweise seine Entscheidungsfindung anpassen und auf kritische Wände unter Druckbeanspruchung reagieren oder die zugrundeliegenden Kanten gar nicht erst in die Zelle einziehen.

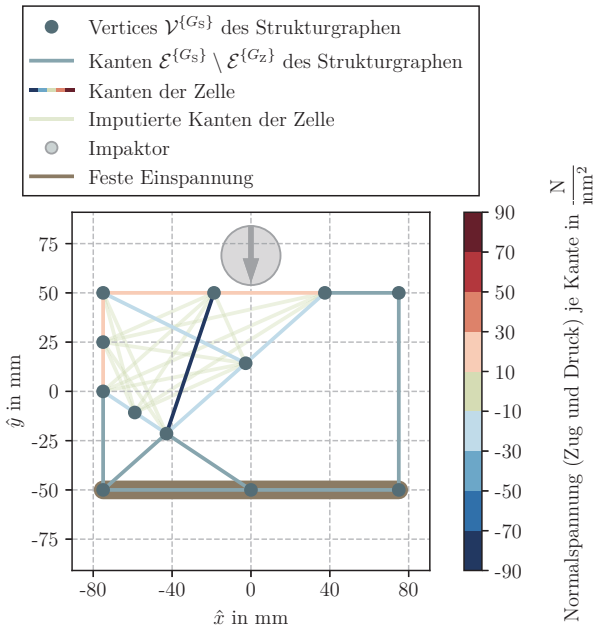
Da die Axial- und Lateralkraft jeweils für jede Kante der Zelle beschrieben werden, werden $|\mathcal{E}_{\text{ges}}^{\{G_{\text{Zm}}\}}|$ Werte im Featurevektor benötigt, welche den Kanten auf dem Rahmen und den Kanten im Adjazenzvektor zugeordnet sind. Es eignet sich, analog zu der inneren Energiedichte, ein Box-Raum zur Beschreibung der Schnittkräfte.

Die Schnittkräfte müssen normiert werden, damit sie zwischen unterschiedlichen Zellen mit verschiedenen Lastniveaus und Querschnittsgeometrien der Wände für den Agenten vergleichbar bleiben. Analog zu der inneren Energiedichte, bei der die Normierung über das Volumen geschieht, werden die Kräfte mit ihrer Wirkfläche geometrisch normiert. Dadurch lassen sich die normierten Schnittkräfte als ein Spannungsmaß, welches die wirkende Last pro Einheitsfläche angibt, auffassen. Die Axialkräfte bewirken eine Normalspannung (Zug bzw. Druck) in der Wand und die Lateralkräfte bewirken eine Schubspannung in der Wand. Somit ist es unabhängig von der vorhandenen Geometrie und vergleichbar mit anderen Zellengeometrien. An einer Beispielzelle werden die ermittelten Normal- und Schubspannungen in Abbildung 5–22 für die eingezogenen Kanten dargestellt.

Folgende Erläuterungen beziehen sich auf die Schnittkräfte, da diese die zentrale Berechnungsgrundlage für die daraus abgeleiteten Spannungen sind. Kraftverläufe in Crashsimulationen sind häufig hochgradig nichtlinear und verrauscht. Deshalb ist es üblich, diese Kraftverläufe zu filtern. Hier wird ein für Crashsimulation häufig genutzter *Channel Frequency Class*-Filter (CFC-Filter) (Society of Automotive Engineers 1995) herangezogen. Als Filtertyp wird der Filter CFC-600 genutzt, welcher hochfrequente Anteile im Signal herausfiltert, ohne dass dabei zu viele Details verloren gehen.

Zur Auswertung der Schnittkräfte muss sich, ebenso wie bei den anderen zeitabhängigen Strukturantworten überlegt werden, zu welchem Zeitpunkt sie ermittelt werden. Aufgrund der Nichtlinearität und mechanischen Komplexität der Kraftverläufe, muss die Wahl des Auswertezeitpunkts bei den Kräften gegenüber der anderen Features besonders akribisch diskutiert werden. Es ist nicht zwingend nötig, dass nur ein Zeitpunkt zur Beschreibung des Kraftverlaufs herangezogen wird. Theoretisch ist es denkbar, mehrere Zeitpunkte für die Observation aufzubereiten. Allerdings befindet man sich in einem Spannungsfeld. Mehr Auswertepunkte bilden das Problem genauer ab. Das bedeutet aber auch, dass der Agent mehr Daten anlernen muss und das Training dadurch schwieriger wird und ggfs. länger dauert. Ein einfacher Ansatz wäre die Bestimmung von äquidistant verteilten Zeitpunkten. Dabei kann es allerdings passieren, dass die entscheidenden Punkte, wie beispielsweise Kraftpeaks, nicht detektiert werden. Ein Ansatz zur adaptiven Bestimmung sinnvoller Auswertezeitpunkte wurde von Triller et al. (2022) für eine Weiterentwicklung der ESLM-bereits vorgestellt. Dort werden die Zeitpunkte so abgetastet, dass dominierenden nichtlineare Effekte in dem Signal abgebildet werden.

Aufgrund der eingebrachten Komplexität und auch Unsicherheit, ob die Betrachtung von Spannungen sinnvoll ist, wird zunächst ein Ansatz verfolgt, der nur einen Zeitpunkt zur Auswertung für jede Wand vorsieht. Eine Möglichkeit ist es, einen repräsentativen Auswertezeitpunkt der gesamten Zelle zu ermitteln. Alle Kräfte würden dann an diesem



(a) Den Kanten zugeordneter Featurevektor der Normalspannungen

(b) Den Kanten zugeordneter Featurevektor der Schubspannungen

Abbildung 5–22: Visualisierung des Spannungs-Features anhand einer Beispielzelle mit zwei bereits eingezogenen Kanten

repräsentativen Zeitpunkt ausgelesen. Das könnte beispielsweise der Zeitpunkt sein, an dem die Gesamtbelastung in der Zelle maximal ist. In Konsequenz können aber relevante Informationen von Kanten, die zu anderen Zeitpunkten eine hohe Last erfahren, verloren gehen. Deshalb wird sich schlussendlich dafür entschieden, dass jede Wand i ihren individuellen und kritischen Auswertezeitpunkt $t_{\mathrm{max}}^{\mathrm{F},i}$ besitzt, die dem maximalen Kraftpeak in dem zugehörigen gefilterten Kraftverlauf entspricht. Der Nachteil ist, dass die Zelle in ihren Schnittkräften nicht mehr zu einem einheitlichen Zeitpunkt bewertet wird. Das muss beispielsweise bei der Kraft- bzw. Spannungsaggregation berücksichtig werden, damit nicht verschiedene Auswertezeitpunkte von verschiedenen Wänden vermischt werden und die mechanische Bedeutung damit verfälscht wird.

Entsprechend ist das Bilden eines Mittelwertes zur Aggregation von den Maximalspannungen ungünstig, da es passieren kann, dass sich Spannungen über verschiedene Teilwände hinweg gegenseitig aufheben. Analog zur Wahl des kritischen Zeitpunkts wird bei der Aggregation deshalb auch die kritischste Teilwand in der Zelle ausgewählt, welche dann repräsentativ für das gesamte Segment aus Kanten und daraus abgeleiteten Wänden sein soll.

Deformationsbild: Das letzte betrachtete Feature ist ein Deformationsbild. Das ist ein Bild vom deformierten Querschnitt der Zelle in der Auswerteebene des Evaluationsgraphen. Dieses Bild wird dann von dem Agenten durch ein CNN analysiert und verarbeitet. Durch die Deformationsbilder soll der Agent einen Gesamteindruck der Deformation der Zelle über die Verschiebungen, welche ansonsten lediglich an der Position der Vertices des Strukturgraphen ermittelt werden, erhalten. Somit lässt sich das Deformationsverhalten netzunabhängig im Bereich zwischen den Vertices des Strukturgraphen beschreiben. Die Rahmenkanten der Zelle in dem Deformationsbild sind dabei stets in gleicher Reihenfolge eingefärbt, um für den Agenten auch in komplexen Deformationszuständen eine Möglichkeit zur Zuordnung von Kanten zu ihrer Ausgangsposition zu bieten und durch die Farbwechsel zu erkennen, wo sich die Vertices befinden, an denen Kanten eingezogen werden. Ein Beispiel für das Deformationsbild ist in Abbildung 5–23 gezeigt.

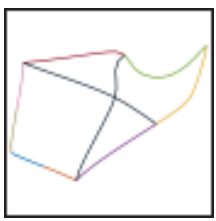


Abbildung 5–23: Darstellung des Deformationsbilds. Der Rahmen um das Bild ist nicht Teil des Features und zeigt lediglich die Abmaße des Bilds

Die Abmaße des Bilds sind 84 px × 84 px. Es besteht aus drei Farbkanälen, was in 21 168 Zahlen zur Beschreibung des Bilds resultiert. Bilder werden in GYM und STABLE-BASELINES3 durch einen Box-Raum beschrieben. Die exakten Abmaße für das Bild stammen aus den Veröffentlichungen von Mnih et al. (2013) und Mnih et al. (2015), die eine CNN-Architektur entwickelt haben, die erfolgreich beim Training von RL-Problemstellungen eingesetzt wurde. Zusätzlich sind die Abmaße ein guter Kompromiss aus nötigem Detailgrad, um lokale Deformationen gut auflösen zu können, und einer für Bilder verhältnismäßig kleinen Auflösung, um nicht mehr Pixel als nötig zur Beschreibung der Deformation nutzen zu müssen. Allerdings ist anzumerken, dass die Anzahl an Trainingsschritten in der Umgebung aus den o. g. Veröffentlichungen mit 50 Millionen Schritten um zwei Größenordnungen größer ist, als es in dieser Dissertation möglich ist. Da aber auch die Charakteristik der Umgebung gänzlich anders ist, wird das Training eines Agenten zeigen, wie viel Mehrwert die zusätzliche Betrachtung von Bildern hat.

Dadurch, dass der Wertebereich für jeden der Farbkanäle von 0 bis 255 bekannt ist, ist keine weitere Normierung nötig. Der Auswertezeitpunkt ist, wie bei dem Verschiebungs-Feature, $t^{\mathrm{IE}}_{\mathrm{max}}$. Zur Imputation wird, wie bereits angesprochen wurde, ein Dummy-Wert von 255 für alle Pixel und Farbkanäle genutzt, sodass, wenn das Simulationsmodell nicht herstellbar ist, ein komplett weißes Bild an den Agenten übergeben wird.

Zusammenfassung: Für eine zusammenfassende Übersicht werden in Tabelle 5–4 alle beschriebenen Features mitsamt deren Eigenschaften dargestellt.

Tabelle 5–4: Übersicht über die Features des Observationsraums

Feature	Subraum	Form	Normierung	Zeitpunkt	Aggregation
Koordinaten	Box	$\lvert\mathcal{V}^{\{G_{\mathrm{Zm}}\}}\rvert \times 2$	/	t_0	/
Verschiebungen	Box	$\lvert\mathcal{V}^{\{G_{\mathrm{Zm}}\}}\rvert \times 2$	GN	$t^{\mathrm{IE}}_{\mathrm{max}}$	/
Kantenlängen	Box	$\lvert\mathcal{E}^{\{G_{\mathrm{Zm}}\}}_{\mathrm{ges}}\rvert$	/	t_0	Summe
Wandstärke	Box	1	/	t_0	/
Gesamtlänge	Box	1	/	t_0	/
Herstellbarkeit	Multibinär	5	/	t_0	/
Zellenanbindung	Multibinär	$\lvert\mathcal{V}^{\{G_{\mathrm{Zm}}\}}\rvert$	/	t_0	/
Adjazenzvektor	Multibinär	$\lvert\mathcal{I}^{\{G_{\mathrm{Zm}}\}}\rvert$	/	t_0	/
Innere Energiedichte	Box	$\lvert\mathcal{E}^{\{G_{\mathrm{Zm}}\}}_{\mathrm{ges}}\rvert$	GN	$t^{\mathrm{IE}}_{\mathrm{max}}$	Mittelwert
Normalspannungen	Box	$\lvert\mathcal{E}^{\{G_{\mathrm{Zm}}\}}_{\mathrm{ges}}\rvert$	GN	$t^{\mathrm{F},i}_{\mathrm{max}}$	Maximum
Schubspannungen	Box	$\lvert\mathcal{E}^{\{G_{\mathrm{Zm}}\}}_{\mathrm{ges}}\rvert$	GN	$t^{\mathrm{F},i}_{\mathrm{max}}$	Maximum
Deformationsbild	Box	$84 \times 84 \times 3$	/	$t^{\mathrm{IE}}_{\mathrm{max}}$	/

5.6 Aufbau und Funktionsweise der Agenten

5.6.1 Auswahl und Beschreibung des Trainingsalgorithmus

Für das Training der Agenten wird der *Proximal Policy Optimization*-Algorithmus (PPO-Algorithmus) (Schulman et al. 2017) in der Implementierung von STABLE-BASELINES3 genutzt. Es handelt sich dabei um einen state-of-the-art Actor-Critic-Algorithmus, welcher sich durch seine Sample-Effizienz und seine guten Performance in verschiedensten Trainingsumgebungen auszeichnet. Die Erklärungen in diesem Abschnitt basieren auf den Ausführungen der Primärliteratur von Schulman et al. (2017) und sind als Erweiterung der RL-Grundlagen in Abschnitt 3.4.4 mit einer konkreten algorithmischen Umsetzung eines RL-Algorithmus zu verstehen.

Das Ziel des PPO-Algorithmus ist die Maximierung der Loss-Funktion

$$L_t^{\mathrm{CLIP+VF+S}}(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t \left[L_t^{\mathrm{CLIP}}(\boldsymbol{\theta}) - c_1 L_t^{\mathrm{VF}}(\boldsymbol{\theta}) + c_2 S\left[\pi_{\boldsymbol{\theta}}\right](s_t) \right]. \tag{5–12}$$

Die Loss-Funktion entspricht einer alternativen, auf den PPO-Algorithmus zugeschnitten Formulierung einer Metrik zur Bewertung der Agentenperformance, wie sie mit Gleichung (3–16) in Abschnitt 3.4.4 eingeführt wurde. Diese setzt sich aus verschiedenen Termen zusammen, welche im Folgenden beleuchtet werden.

CLIP-Term L_t^{CLIP}: Der zentrale Term, der die Verbesserung der Strategie steuert und somit für das Training des Actors zuständig ist, nennt sich CLIP-Term $L_t^{\mathrm{CLIP}}(\boldsymbol{\theta})$. Berechnet wird $L_t^{\mathrm{CLIP}}(\boldsymbol{\theta})$ nach der Formel

$$L_t^{\mathrm{CLIP}}(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t \left[\min\left(\xi_t\left(\boldsymbol{\theta}\right) \hat{A}_t; \mathrm{clip}\left(\xi_t\left(\boldsymbol{\theta}\right); 1-\epsilon; 1+\epsilon\right) \hat{A}_t \right) \right]. \tag{5–13}$$

Zur Verbesserung der Strategie wird eine Schätzung der Advantage-Funktion $\hat{A}_t$ herangezogen. Allgemein liefert die Advantage-Funktion $A(s,a) = Q(s,a) - V(s)$ einen Wert, der angibt, wie viel besser die gewählte Aktion im Vergleich zu dem Mittelwert über alle Aktionen ist. Da die Berechnung der Aktionswertfunktion $Q(s,a)$ in praktischen Aufgabenstellungen durch die Komplexität der Umgebung schwierig ist, wird die Advantage-Funktion durch den TD-Fehler $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ (vgl. Gleichung (3–11)) geschätzt. Diese Schätzung unterliegt allerdings einem Bias, da sie nur von einem Sample abhängt. Gleichzeitig besitzt diese Schätzung aus dem gleichen Grund eine geringe Varianz. Für den PPO-Algorithmus wird daher häufig das sog. *Generalized Advantage Estimate* (GAE) (Schulman et al. 2015) genutzt, welche eine Schätzung über mehrere Samples ermöglicht:

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}. \tag{5–14}$$

γ ist der bekannte Diskontierungsfaktor und λ ist ein durch das GAE neu eingeführter Hyperparameter, welcher typischerweise zwischen 0 und 1 gewählt wird und den Trade-Off bzgl. Bias und Varianz der Schätzung steuert.

Das Verhältnis aus Wahrscheinlichkeiten der aktuellen Strategie und einer älteren Strategie $\xi_t(\boldsymbol{\theta})$[1] berechnet sich über

$$\xi_t(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}\left(a_t \mid s_t\right)}{\pi_{\boldsymbol{\theta}_{\text{old}}}\left(a_t \mid s_t\right)}, \quad \text{sodass} \quad \xi_t(\boldsymbol{\theta}_{\text{old}}) = 1. \tag{5–15}$$

Das Verhältnis bemisst, wie stark sich die Wahrscheinlichkeit eine Aktion zu wählen nach einem Update der Strategie ändert, wenn Aktion a_t in Zustand s_t gewählt wird. Durch die Multiplikation von ξ_t mit $\hat{A}_t$ werden beispielsweise wenig sinnvolle Aktionen ($\hat{A}_t < 0$) besonders bestraft, wenn $\xi_t > 1$ ist, d. h., wenn die geupdatete Strategie eine höhere Wahrscheinlichkeit hat, diese Aktion zu wählen als die alte Strategie.

Damit sich die Strategie nicht zu stark zwischen den Updates ändert, wird der Hyperparameter ϵ eingeführt. Dieser grenzt ein, wie stark sich die Strategie ändern darf, indem das Verhältnis ξ_t zwischen $[1-\epsilon, 1+\epsilon]$ begrenzt wird. Dadurch wird die Stabilität des Trainings verbessert. Das ist insbesondere vor dem Hintergrund der in dieser Dissertation vorgestellten Trainingsumgebung sinnvoll, da diese auf den hochgradigen Nichtlinearitäten der Crashmechanik beruht. Kleine Änderungen in der Strategie können schon ein signifikant anderes strukturmechanisches Verhalten erzeugen.

Zusammenfassend werden mit Gleichung (5–13) die Parameter $\boldsymbol{\theta}$ der Strategie so optimiert, dass die Wahrscheinlichkeit eine Aktion mit positivem Advantage-Wert zu wählen maximal ist. Dabei werden die Updates der Strategie begrenzt, um ein stabiles Training zu gewährleisten.

VF-Term L_t^{VF}: Der Unterschied zwischen der vom Critic vorhergesagten Zustandswertfunktion $V_{\boldsymbol{\theta}}(s_t)$ und den durch die Episoden gesammelten wahren Zustandswerten V_t^{targ} wird durch $L_t^{\text{VF}}(\boldsymbol{\theta})$ beschrieben. Das Maß berechnet sich über die Formel

$$L_t^{\text{VF}} = (V_{\boldsymbol{\theta}}(s_t) - V_t^{\text{targ}})^2. \tag{5–16}$$

Dadurch, dass der Term minimiert werden soll, geht er in die Loss-Funktion (5–12) mit negativem Vorzeichen ein. Über den Faktor c_1 kann der/die Anwender*in steuern, wie groß der Beitrag des VF-Terms in der Loss-Funktion ist.

[1] Von der Bezeichnung r_t, welche von Schulman et al. (2017) anstelle von ξ_t für das Verhältnis genutzt wurde, wird abgesehen, um eine Verwechslung mit dem Reward r_t zu vermeiden.

Entropieterm S: Der Entropie-Bonus S generiert, je nach Wahl des Faktors c_2, einen Beitrag zur Loss-Funktion (5–12). Die Strategie ist eine Wahrscheinlichkeitsverteilung über Aktionen. Es kann passieren, dass diese Wahrscheinlichkeitsverteilung während des Trainings kollabiert, die Strategie also damit frühzeitig gegen ein lokales Optimum konvergiert und beispielsweise stets die gleiche Aktion vorhergesagt wird. Ziel ist es durch den Entropie-Bonus, die Strategie dazu zu ermutigen, den Aktionsraum weiterhin zu explorieren, potentiell bessere Strategien zu finden und nicht in einem lokalen Optimum stecken zu bleiben. Die Entropie S für die diskrete Verteilungsfunktion $\pi_{\boldsymbol{\theta}}$ lässt sich berechnen über den Zusammenhang

$$S\left[\pi_{\boldsymbol{\theta}}\right](s_t) = -\sum_{a \in \mathcal{A}} \pi_{\boldsymbol{\theta}}\left(a \mid s_t\right) \log_2\Big(\pi_{\boldsymbol{\theta}}\left(a \mid s_t\right)\Big). \tag{5–17}$$

5.6.2 Trainingsschleife

Die im vorigen Abschnitt eingeführte Zielfunktion (vgl. Gleichung (5–12)) ist die Grundlage für das Training des Agenten. Bei dem PPO-Algorithmus handelt es sich um einen On-Policy-Algorithmus. Dabei werden die Trainingsdaten online basierend auf der aktuellen Strategie des Agenten generiert. Die zugrundeliegende Trainingsschleife wird in Abbildung 5–24 dargestellt.

Die Trainingsschleife zeigt den prinzipiellen Ablauf des Trainings von einem untrainierten zu einem trainierten Agenten. Zentraler Bestandteil ist der sog. *Rollout-Buffer*. Ein Buffer (deutsch: *Puffer*) ist ein Zwischenspeicher für Daten. In dem Rollout-Buffer werden Aktionen, zugehörige standardisierte Observationen, Rewards, Terminierungssignale und Advantages für jeden Schritt in der Trainingsumgebung gespeichert. Zusätzlich wird, sofern erforderlich, ein Reset der Umgebung durchgeführt. Da dies aber nicht direkt Einfluss auf das Training hat, werden die Resets hier nicht spezifisch gezeigt. Sie dienen lediglich zur Formulierung einer neuen Datengrundlage. Es werden so lange Daten im Buffer gesammelt, bis dieser gefüllt ist. Wie viele Samples in den Buffer passen, ist ein von dem/der Nutzer*in festzulegender Hyperparameter n_{steps} für das Training des Agenten. Wichtig ist, dass alle gesammelten Daten im Buffer auf der aktuellen Strategie des Agenten basieren. Wenn der Buffer voll ist, wird der Agent geupdatet. Das Updaten basiert auf den im vorigen Abschnitt gezeigten Formeln. Das Training ist vollendet, wenn eine festgelegte Anzahl an Trainingsschritten durchgeführt wurde.

Um das Update des Agenten bzw. seiner Strategie im Detail zu erklären, werden im Folgenden zwei zentrale Begrifflichkeiten aus dem Training von NNs im Kontext des RLs eingeführt:

- Als *Epoche* wird der gesamte Durchlauf der im Rollout-Buffer befindlichen Trainingsdaten durch die NNs des Agenten bezeichnet. Die Anzahl an Epochen n_{epochs} zum Training wird von dem/der Anwender*in festgelegt.

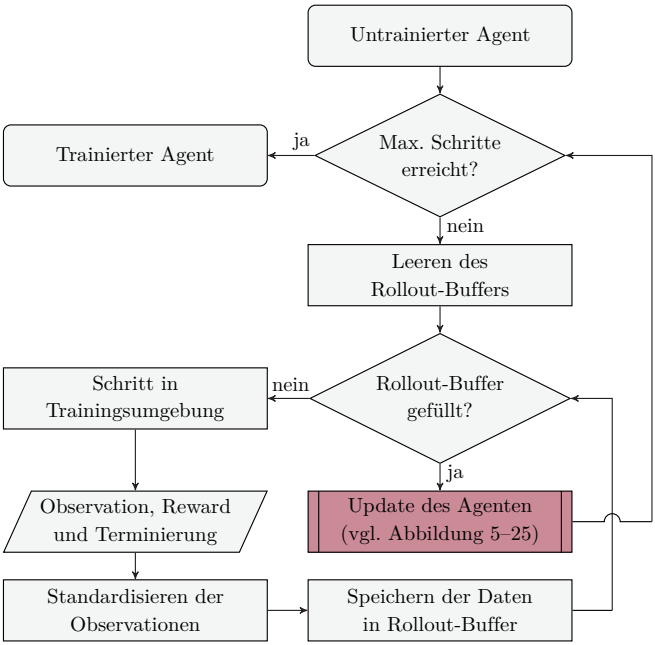


Abbildung 5–24: Trainingsschleife für On-Policy-Algorithmen (modifiziert aus Trilling et al. (2022a))

- Der Datensatz wird in einzelne *Batches* der Größe n_{batches} eingeteilt, welche nacheinander abgearbeitet werden. Ein Batch ist eine Teilmenge der Trainingsdaten, die für eine einzelne Aktualisierung der Modellparameter verwendet wird.

Der Ablauf des Agentenupdates wird in Abbildung 5–25 dargestellt. Aus dem Batch werden zunächst die gespeicherten Advantages ausgelesen und, sofern von dem/der Nutzer*in angefordert, für den Batch normalisiert. Das reduziert die Varianz in den Schätzungen der Advantages, was sich positiv auf das Training ausüben kann.

Anschließend wird auf Basis von Gleichung (5–12) der zugehörige Zielfunktionswert berechnet. Damit können wiederrum die Gradienten durch den Backpropagation-Algorithmus bestimmt werden, die bestimmen, wie stark die Parameter des dem Agenten zugrundeliegenden NNs angepasst werden müssen. Wenn die Gradienten zu groß werden, spricht man vom Phänomen der *explodierenden Gradienten*, welche sich stark negativ auf das Training des Agenten auswirken. Es kann ggfs. zu einer übermäßigen Anpassung der Strategie mit unkontrollierten Auswirkungen kommen. Um das zu vermeiden, wird die Gradienten-Norm geclipped (Pascanu et al. 2012). Dafür muss von dem/der Nutzer*in ein Grenzwert festgelegt werden, ab dem der Gradient als zu groß gilt.

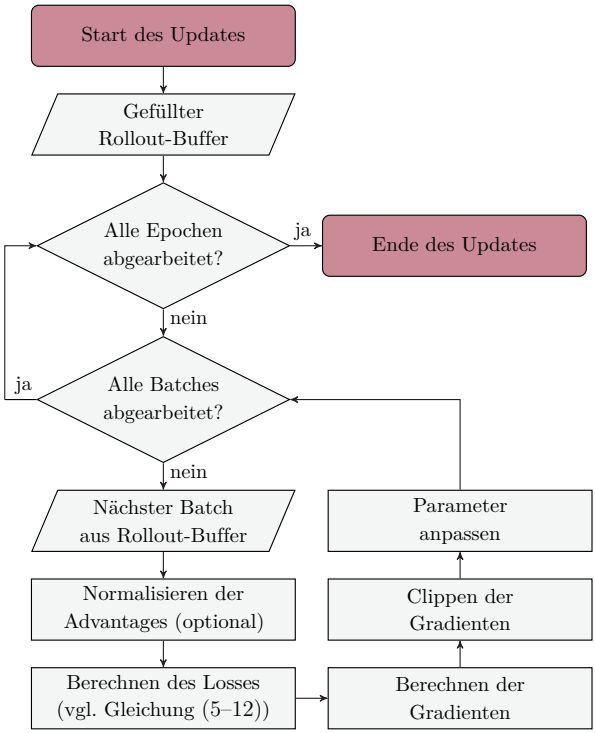


Abbildung 5–25: Update des Agenten mit dem PPO-Algorithmus basierend auf der Implementierung von STABLE-BASELINES3

Schlussendlich können die Strategieparameter auf Basis eines Optimierungsalgorithmus und den ermittelten Gradienten angepasst werden. Als Optimierungsalgorithmus zur Anpassung der Strategieparameter wird der *Adaptive Moment Estimation*-Algorithmus (Adam-Algorithmus) (Kingma und Ba 2015) genutzt, welcher in STABLE-BASELINES3 als Standardoptimierer für den PPO-Algorithmus hinterlegt ist. Adam ist eine Erweiterung der stochastischen Gradientenverfahren, der eine individuelle und adaptive Lernrate für jedes Gewicht des NNs bestimmt. Dabei wird Parametern mit hoher Varianz in den Gradienten eine kleinere Lernrate zugewiesen. Gleichzeitig werden die Updates der Parameter aus einem Mittelwert aus vorangegangenen Gradienten berechnet, um sprunghafte Änderungen der Parameter zu vermeiden. Adam ist laut den Autoren u. a. recheneffizient, hat geringe Speicheranforderungen und eignet sich für Probleme mit großen Daten- und Parametermengen.

5.6.3 Standardisieren der Observationen

Alle Features, deren Wertebereich für den Agenten unbekannt sind und die über einen Box-Raum definiert sind, werden über die Transformation

$$\tilde{x} = \text{clip}\left(\frac{x - \overline{x}_t}{s_{\text{emp},t}}; -3; 3\right) \tag{5–18}$$

auf einen Mittelwert von 0 und eine Standardabweichung von 1 basierend auf dem empirischen Mittelwert $\overline{x}_t$ und der empirischen Standardabweichung $s_{\text{emp},t}$ der Ausgangsverteilung für betrachtete Werte x standardisiert. Das Clipping dient dazu, die Robustheit des Agenten in der späteren Anwendung zu gewährleisten. Stärkere Ausreißer von standardisierten Observationen, welche im Training nicht abgebildet werden, können so abgefangen werden und zurückgeführt werden auf einen Entscheidungsbereich, in dem der Agent trainiert worden ist. Das soll ungünstige Entscheidungen des Agenten verhindern.

Durch die Standardisierung einiger Features sind diese in einer gleichen Größenordnung, was einen positiven Einfluss auf die Konvergenz und die numerischen Stabilität des Trainings hat. In einem Training, bei dem Features nicht die gleiche Größenordnung hätten, würde ein gradientenbasierter Optimierungsalgorithmus primär versuchen, die Features mit größerer Ordnung anzupassen, da diese den größten Einfluss auf die Loss-Funktion haben. Das resultiert in langsamerer Konvergenz und ggfs. schlechteren lokalen Optima. Features, die nicht standardisiert werden, liegen naturgemäß in der gleichen Größenordnung wie die standardisierten Features und benötigen daher keine zusätzliche Normierung.

Im Gegensatz zum SLs, wo der gesamte Datensatz vor dem Training zur Verfügung steht und dieser entsprechend direkt standardisiert werden kann, werden die Datenpunkte beim RL erst nach und nach während des Trainings generiert. Deshalb müssen der Mittelwert und die Standardabweichung iterativ während des Trainings bestimmt werden. Die Implementierung der iterativen Bestimmung der statistischen Kenngrößen zur Standardisierung – das ist der empirische Mittelwert $\overline{x}_t$ und die empirische Standardabweichung $s_{\text{emp},t}$ – in STABLE-BASELINES3 orientiert sich an Welfords Algorithmus (Welford 1962). Es ist wichtig anzumerken, dass das Standardisieren der Features außerhalb der Umgebung und während des Trainings geschieht und nicht Bestandteil der Umgebung ist. Jeder Eintrag eines Featurevektors wird einzeln standardisiert und erhält somit seinen eigenen statistischen Kennwert.

Durch das iterative Updaten des Mittelwerts und der Standardabweichung während des Trainings tritt ein Phänomen namens *Covariate Shift* oder auch *Data Drift* auf, welches dazu beiträgt, dass die Trainingsumgebung nicht-stationär ist. Dieses Phänomen beschreibt den Drift der Datenverteilung über die Trainingsschritte, insbesondere zu Beginn des Trainings, wenn die Approximationen des Mittelwerts und der Standardabweichung aufgrund der geringen Anzahl an Samples noch nicht präzise genug ist. Aus

dem *Gesetz der großen Zahlen* folgt bei steigender Anzahl betrachteter Samples, dass sich die statistischen Kenngrößen an die zugehörigen theoretischen Werte μ_t bzw. σ_t der Population bzw. Datenverteilung annähern. Zwar erschwert der Drift das Training des Agenten, aber grundsätzlich können viele RL-Algorithmen mit solchen nicht-stationären Phänomenen umgehen.

5.6.4 Konzeptionelle Architektur der Agenten

In diesem Abschnitt wird die Architektur der Agenten beschrieben. Da zu diesem Zeitpunkt noch unbekannt ist, welche Features Bestandteil des Observationsraums im finalen Agenten sind, welche Einstellungen für die dem Agenten zugrundeliegende Strategie verwendet werden und für welche Anzahl an Zellseiten $|\mathcal{S}^{\{G_{\mathrm{Zm}}\}}|$ der Agent überhaupt trainiert wird, wird hier ein konzeptioneller Aufbau der Agenten vorgestellt. In der Implementierung von STABLE-BASELINES3 besteht die Strategie eines Agenten aus einem Feature-Extractor und zwei dahinter geschalteten MLPs für den Actor und den Critic.

Die Aufgabe des Feature-Extractors ist es, aus komplexen oder unstrukturierten Eingangsdaten wie Bildern oder zeitabhängigen Daten, wesentliche Merkmale der Daten in Form einer vektoriellen Einbettung zu finden. Für die hier implementierten Agenten ist das nur für die Deformationsbilder nötig, welche durch das sog. Nature-CNN (Mnih et al. 2013; Mnih et al. 2015) vektoriell eingebettet werden. Das Nature-CNN besteht aus aus drei Convolution-Layern ohne Padding und ReLU-Aktivierung, welche in Tabelle 5–5 dargestellt werden.

Tabelle 5–5: Architektur der Convolution-Layer im Nature-CNNs in der Implementierung von STABLE-BASELINES3

Layer	# Eingangschannel	# Ausgangschannel	Kernelgröße	Stride
1	3	32	(8×8)	4
2	32	64	(4×4)	2
3	64	64	(3×3)	1

Nach den Convolution-Layern folgt der Head des CNNs. Dieser nimmt 3136 Werte entgegen, welche für das Deformationsbild mit der Form $84 \times 84 \times 3$ nach den Convolution-Layern anfallen. Diese Anzahl ergibt sich aus der Größe des Bildes und der Architektur der Convolution-Layer aus Tabelle 5–5. Der Head reduziert die 3136 Werte auf eine vektorielle Einbettung der Dimension 256 ohne Hidden-Layer und mit ReLU-Aktivierung für die Outputs. Die Dimension von 256 wird gewählt, da es sich um den Standardwert in STABLE-BASELINES3 handelt und somit in der Anwendung in Kombination mit dem Nature-CNN erprobt ist.

Alle anderen Features, welche strukturiert bzw. tabellarisch vorliegen, werden, wenn nötig, zeilenweise konkateniert, sodass diese ebenfalls als Vektor vorliegen. Ein Beispiel hierfür ist das Koordinaten-Feature, welches für jeden Vertex auf dem Zellrahmen (Zeilen) eine Vertexposition in $\hat{x}$- und $\hat{y}$-Richtung (Spalten) zuweist.

Zuletzt werden im Feature-Extractor alle Vektoren aller Features zu einem Input für die MLPs konkateniert. Dieser Vektor nennt sich Featurevektor und repräsentiert die Observation vollständig in vektorieller Form und eignet sich damit für die weitere Verarbeitung in den MLPs. Das Actor-MLP generiert aus dem Featurevektor eine Wahrscheinlichkeitsverteilung über Aktionen und das Critic-MLP bestimmt den Zustandswert der betrachteten Observation.

In Abbildung 5–26 wird die gesamte Architektur des Agenten qualitativ gezeigt. Das beinhaltet entsprechend auch die konzeptionelle Architektur des Actor- und des Critic-MLPs. Wie bereits angesprochen wurde, ist die exakte Architektur eines Agenten abhängig von den genutzten Features der Observation, den gewählten Hyperparametern für die MLPs und der Anzahl an Zellseiten $|\mathcal{S}^{\{G_{\mathrm{Zm}}\}}|$, für die der Agent trainiert wird. Konkrete Parameter für die Anzahl an Hidden-Layern, Anzahl an Neuronen in jedem der Layer und der Aktivierungsfunktion in den Neuronen der Hidden-Layer werden in Abschnitt 5.7.6 festgelegt. Dort wird ebenfalls festgelegt, nach welcher Methode die initialen, zufällig generierten Gewichte der zugrundeliegenden NNs bestimmt werden.

Die Länge des Featurevektors und damit die Anzahl der Neuronen im Input-Layer des Actor- und Critic-MLPs ergibt sich aus der Summe der Einbettungsgröße der einzelnen Features, welche sich direkt aus der in Tabelle 5–4 aufgeführten Form ableiten. Eine Ausnahme ist hier das Deformationsbild. Für dieses ist für die Länge des Featurevektors nicht die originale Bildgröße aus der Tabelle entscheidend, sondern die vektorielle Einbettung durch das CNN-bestehend aus 256 Werten.

Der Output des Actor-MLP ist ein linearer Layer und hat die Dimension $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}| + 2$. Die $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|$ Neuronen steuern, welche Kante aktiviert wird und die zusätzlichen zwei Neuronen steuern, ob der Agent die Episode terminiert oder nicht. Der Layer gibt die Logits $\boldsymbol{z}$ der Aktionen zurück. Logits sind als roher Output des MLPs zu verstehen und werden als Input zur Generierung einer multikategorischen Wahrscheinlichkeitsverteilung genutzt. Diese setzt sich zusammen aus einer kategorischen Verteilung für die einzubringende Kante und einer weiteren kategorischen Verteilung, ob die Episode zu terminieren ist. Auf die zu den Logits zugehörigen Wahrscheinlichkeiten einer kategorischen Verteilungsfunktion kann über die *Soft-Max-Funktion*

$$\varphi(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{5–19}$$

geschlossen werden. Dabei entspricht N der Anzahl der Neuronen des Output-Layers. Die Normierung mit $\sum_{j=1}^{N} e^{z_j}$ stellt sicher, dass die Summe aller $\varphi(\boldsymbol{z})_i$ genau 1 ergibt, wie es für eine Wahrscheinlichkeitsverteilung gefordert ist.

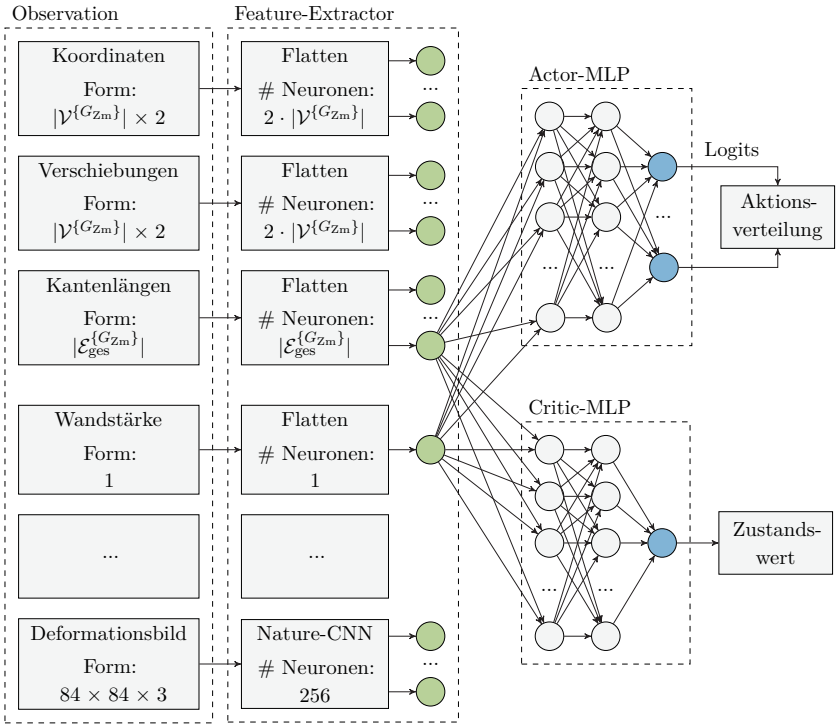


Abbildung 5–26: Konzeptionelle Architektur der Agenten. Zur Übersichtlichkeit sind nicht alle Verbindungen zwischen dem Featurevektor und dem Actor bzw. Critic eingezeichnet. Das Actor- und Critic-MLP sind exemplarisch mit zwei Hidden-Layern dargestellt

Während des Trainings wird die zu wählende Aktion aus dieser Verteilung gesampelt, wodurch die Exploration des Agenten naturgemäß gefördert wird. In der Anwendung des Agenten kann dann stets die Aktion mit höchster Wahrscheinlichkeit für eine sinnvolle Strukturversteifung gewählt werden. Für tiefergehende Informationen über die kategorische Verteilung hat Murphy (2012) eine detailreiche Übersicht mit ML-Fokus zusammengestellt. Dort ist die kategorische Verteilung als Spezialfall der „Multinomialverteilung“ zu finden. Der Output-Layer bei dem Critic-MLP ist ebenfalls ein linearer Layer, welcher aber nur aus einem Neuron besteht. Dieses Neuron soll den Zustandswert der betrachteten Observation ausgeben. Hier sind keine weiteren Anpassungen an dem Output nötig.

5.7 Training der Agenten

5.7.1 Vorgehen

Für die neue Heuristik sollen Zellen mit drei, vier und fünf Seiten unterstützt werden, um möglichst viele verschiedene Strukturen und Zellen anpassen zu können. Zur Auswahl geeigneter Features und Hyperparameter für das Training der Agenten werden Parameterstudien durchgeführt.

Das Training der zugehörigen Agenten wird in drei Stufen unterteilt, welche in Abbildung 5–27 dargestellt werden. Entscheidend ist, dass dieses mehrstufige Training der Agenten nur einmal gemacht werden muss und der Agent dann in der Heuristik für verschiedenste Optimierungsaufgaben genutzt werden kann.

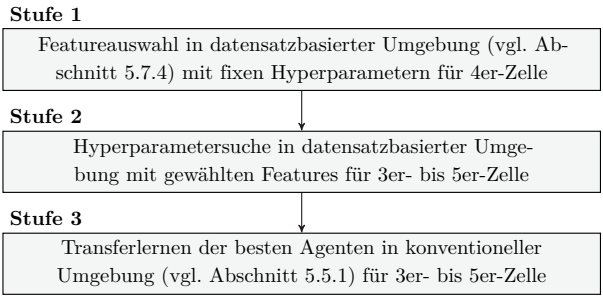


Abbildung 5–27: Stufen des Trainings der Agenten

In *Stufe 1* wird an einer 4er-Zelle für fixe Hyperparameter des Agenten systematisch untersucht, welche Features aus Tabelle 5–4 relevant für das Training des Agenten sind. Dazu wird eine *datensatzbasierte* Variante der vorgestellten Umgebung implementiert. Der in Abschnitt 5.5.1 vorgestellte Ablauf der Umgebung wird dafür abgewandelt und kommt erst in Stufe 3 in unmodifizierter Form zum Einsatz. Die datensatzbasierte Umgebung ermöglicht es, die Agenten sehr zeiteffizient zu trainieren. Die datensatzbasierte Trainingsumgebung basiert dabei auf der bisher vorgestellten Umgebung, ist aber so verändert, dass keine Simulationen mehr während des Trainings ausgeführt werden müssen. Deshalb eignet sich diese Herangehensweise gegenüber dem Ablauf aus Abschnitt 5.5.1, da so viele Agenten effizient zur Evaluation von Features und Hyperparametern trainiert werden können.

Damit während des Trainings auf die Simulation von Strukturen verzichtet werden kann, müssen diese vorab berechnet werden, um auf die Strukturantworten der Modelle schließen zu können und das Formabweichungsmaß bestimmen zu können. Der Datensatz besteht dabei aus ca. 16 000 individuellen Zellen und zugehörigen Lastfällen, für die alle

Kantenkombinationen für eine maximale Anzahl einziehbarer Kanten vorab simuliert werden. Für jeden dieser Datenpunkte ist die Observation abgespeichert und kann in der datenbasierten Umgebung direkt eingelesen werden, ohne eine aufwändige FE-Simulation abwarten zu müssen. Der Nachteil an dieser Herangehensweise ist, dass die mechanische Diversität der Daten auf den Datensatz begrenzt ist.

In *Stufe 2* wird mit den gewählten Features untersucht, welche Hyperparametereinstellungen für die Agenten und den PPO-Algorithmus der 3er- bis 5er-Zelle geeignet sind. Hierfür wird ebenfalls die datenbasierte Umgebung herangezogen. Theoretisch interagieren die Features aus Stufe 1 mit den Hyperparametern aus Stufe 2, weshalb eine getrennte Analyse der Parameter die Wechselwirkung beider nicht abbilden kann. Das resultiert in einer möglicherweise etwas schlechteren Performance der Agenten. Dieses Vorgehen wird trotzdem gewählt, um einen klar separierten Einfluss zwischen der Wahl von Features und der Wahl der Hyperparametereinstellungen detektieren zu können.

Die besten gefundenen Agenten werden in *Stufe 3* weiter trainiert. Grundlage dafür ist die konventionelle, simulationsbasierte und in diesem Kapitel bisher detailliert vorgestellte Umgebung. Diese generiert unabhängig eines Datensatzes weitere, randomisierten Zellverhalten und damit eine erweiterte Wissensgrundlage für die Agenten. Zusätzlich ermöglicht diese Herangehensweise, dass in die Zellen auch drei Kanten eingebracht werden können. Wenn ein bereits auf eine bestimmte Aufgabe trainierter Agent für eine neue, ähnliche Aufgabe weiter trainiert wird, spricht man von Transferlernen.

5.7.2 Datensatzgenerierung

Zur Generierung des Datensatzes für die datenbasierte Trainingsumgebung wird die in Abschnitt 5.5 implementierte, simulationsbasierte Umgebung herangezogen. Alle benötigten Schnittstellen wurden bereits in Abbildung 5–3 vorgestellt.

Der Trainingsdatensatz besteht, wie bereits angesprochen wurde, für jeweils 3er- bis 5er-Zellen aus ca. 16 000 Samples. Bei dieser Anzahl an Samples kann man sich eine ausreichend große Datenlage bei moderatem Rechenaufwand zur Generierung des Datensatzes versprechen. Zur Evaluation und zum Testen aller Agenten wird zusätzlich jeweils ein Datensatz aus 64 Samples generiert.

Ein Sample umfasst die aus den Simulationsdaten extrahierten Observationen für alle Kombinationen an vom Agenten maximal $|\mathcal{I}_{\text{max}}^{\{G_{\text{Zm}}\}}|$ eingezogenen Kanten. Dazu kommt die zugehörige initiale Observation einer Zelle ohne eingezogene Kanten für ein konkretes randomisiertes Modell und einen konkreten Lastfall. Zur Generierung einer Observation in einem Sample ist die Reihenfolge zum Einziehen der Kanten in die Zelle irrelevant. Das mechanische Verhalten der final betrachteten Zelle ändert sich nicht in Abhängigkeit der Einziehreihenfolge der Kanten.

Besondere Aufmerksamkeit benötigt die Berechnung des Rewards. Da der Reward durch das Clipping in Gleichung (5–7) nicht unabhängig von der Reihenfolge des Einbringens von Kanten ist, wird dieser nicht direkt abgespeichert, sondern die zugrundeliegenden Formabweichungswerte, aus denen dann der Reward in der datenbasierten Umgebung nach Formel (5–8) während des Trainings berechnet werden kann.

Die Anzahl an erzeugbaren Kantenkombinationen mit maximal $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}|$ vom Agenten einziehbaren Kanten und $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|$ verfügbaren Positionen zum Einziehen von Kanten ergibt sich für diesen Ansatz aus

$$|\mathcal{I}_{\mathrm{komb}}^{\{G_{\mathrm{Zm}}\}}| = \sum_{k=0}^{|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}|} \binom{|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|}{|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| - k}. \tag{5–20}$$

Von allen in der modifizierten Zelle theoretisch einziehbaren Kanten $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|$ werden $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| - k$ gewählt. Das sind für $k = 0$ alle maximal erlaubten vom Agenten einziehbaren Kanten. Für den Fall $k = |\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| - 1$ darf der Agent nur noch eine Kante einbringen. Auch diese Fälle sind Bestandteil eines Samples, da eine Episode auch schon nach mindestens einer eingebrachten Kante terminieren kann. Wenn $k = |\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}|$ und folglich $\binom{|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|}{0} = 1$ ist, darf der Agent gar keine Kante einziehen, was hier, stellvertretend für die Simulation der Struktur mit leerer Zelle, auch als eine konkrete Kantenkombination aufgefasst wird.

Tabelle 5–6 gibt eine Übersicht über die Anzahl an Kantenkombinationen für verschiedene Zellarten und verschiedene maximale Kantenanzahlen $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}|$ in der Zelle. Dadurch, dass nicht jede Kantenkombination herstellbar ist, korrespondiert die Anzahl der Kantenkombinationen nicht direkt mit der Anzahl der durchzuführenden Simulationen. Die genannten Anzahlen beziehen sich auf ein einzelnes Sample.

Tabelle 5–6: Anzahl an Kantenkombinationen $|\mathcal{I}_{\mathrm{komb}}^{\{G_{\mathrm{Zm}}\}}|$ für eine maximale Anzahl an einziehbaren Kanten $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}|$ in Abhängigkeit der Zellenart und der damit verbundenen Anzahl an möglichen Positionen einer Kante $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}|$ je Sample

Zellenart	$\lvert\mathcal{I}^{\{G_{\mathrm{Zm}}\}}\rvert$	$\lvert\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}\rvert$	$\lvert\mathcal{I}_{\mathrm{komb}}^{\{G_{\mathrm{Zm}}\}}\rvert$
3er-Zelle	6	2	22
		3	42
4er-Zelle	16	2	137
		3	697
5er-Zelle	30	2	466
		3	4526

Aufgrund der signifikant höheren Anzahl an benötigten Kantenkombinationen bei $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| = 3$ wird der Trainingsdatensatz mit $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| = 2$ generiert. Bei 16 000 Samples entspricht das bei der 5er-Zelle bereits weit über 7 Millionen zu simulierende Strukturen,

sofern alle Modelle herstellbar sind. Durch geeignete Parallelisierung lässt sich diese Menge an Simulationen bei einer Rechenzeit des Modells von ca. 5 s in ungefähr einer Woche durchführen.

Die Evaluations- und Testdatensätze werden hingegen mit $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| = 3$ generiert. Für das Training in Stufe 1 und Stufe 2 ist allerdings nur ein Evaluations- und Testdatensatz mit maximal zwei Kanten in der Zelle nötig. Der Vorteil liegt aber darin, dass in Stufe 3, welche auf das Einbringen von drei Kanten angewiesen ist, die gleiche Datengrundlage zur Evaluierung vorliegt, wie in Stufe 1 und 2. So kann verglichen werden, ob die Performance der Agenten durch Stufe 3 wirklich gestiegen ist.

5.7.3 Datensatzanalyse

Bisher wurde der Ressourcenaufwand und die Komplexität der Problemstellung in Abhängigkeit der maximalen Anzahl vom Agenten einzuziehender Kanten in die Zelle diskutiert. In diesem Abschnitt wird untersucht, welchen mechanischen Vorteil mehr Kanten gegenüber einer oder zwei einziehbaren Kanten hat. Das erlaubt auch eine Abschätzung über den mechanischen Effekt für das Einziehen weiterer Wände in die Struktur.

Zur genaueren Beleuchtung wird exemplarisch der Evaluations- und Testdatensatz der 3er- bis 5er-Zellen herangezogen, da dieser mit $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| = 3$ generiert wurde. Für diese Datensätze wird über alle 64 Samples iteriert und für jedes Sample die bestmögliche Kantenkombination identifiziert. Das ist leicht möglich, da alle Observationen und Formabweichungswerte für jede Kantenkombination vorberechnet vorliegen und der Return daraus schnell berechnet werden kann. Die Untersuchung wird je Datensatz dreimal durchgeführt mit $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| = 1$ bis $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| = 3$.

Durch einen Vergleich des undiskontierten Returns $\overline{g}_{0\gamma=1}$ (vgl. Gleichung (3–4)) auf Basis des Rewards r_{t+1} (vgl. Gleichung (5–8)) im Mittel über alle 64 Samples lässt sich erkennen, wie viel es bringt, dass der Agent mehrere Kanten in die Zelle einbringen darf. Für einen fairen Vergleich der Zellperformance darf die Bestrafung zum Einbringen von Kanten p_{ins} nicht Bestandteil des Rewards sein, da dieser so künstlich kleiner wird, wenn mehrere Kanten eingebracht werden. Entsprechend wird ein unbestrafter Reward $r_{t+1}^{(\star)}$ eingeführt, welcher sich analog zu Gleichung (5–8) berechnet, ohne dabei die Bestrafung p_{ins} zu beinhalten. Daraus lässt sich nach Gleichung (3–4) der unbestrafte Return $g_t^{(\star)}$ ermitteln. Tabelle 5–7 stellt den bestmöglichen, mittleren, undiskontierten, und bestraften Return $\overline{g}_{0\gamma=1,\max}$ und unbestraften Return $\overline{g}_{0\gamma=1,\max}^{(\star)}$ bezogen auf den Ausgangszustand der Episode in Abhängigkeit des Zelltypen und der maximalen Anzahl an einziehbaren Kanten für den Evaluations- und Trainingsdatensatz dar.

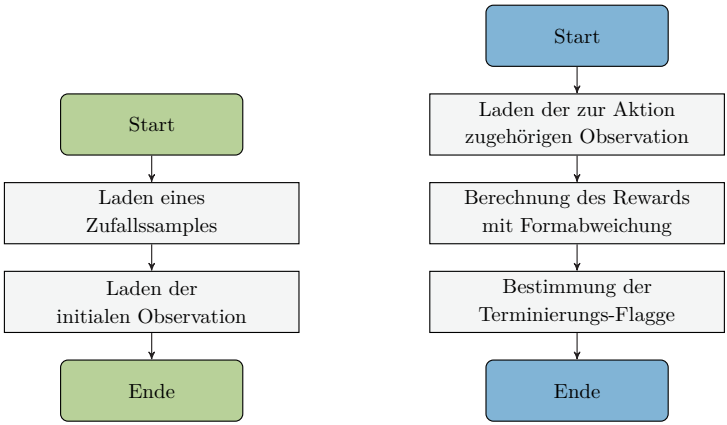
Tabelle 5–7: Bestmöglicher, mittlerer und undiskontierter Return $\overline{g_0}_{\gamma=1,\max}$ und bestmöglicher, mittlerer, undiskontierter und unbestrafter Return $\overline{g_0}^{(\star)}_{\gamma=1,\max}$ für eine maximale Anzahl an einziehbaren Kanten $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}|$ zur Zellversteifung in Abhängigkeit des Zelltyps über den Evaluationsdatensatz und Testdatensatz aus 64 Samples

Zellenart	$\lvert\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}\rvert$	Evaluationsdatensatz		Testdatensatz	
		$\overline{g_0}_{\gamma=1,\max}$	$\overline{g_0}^{(\star)}_{\gamma=1,\max}$	$\overline{g_0}_{\gamma=1,\max}$	$\overline{g_0}^{(\star)}_{\gamma=1,\max}$
3er-Zelle	1	0,576	0,676	0,555	0,655
	2	0,634	0,771	0,606	0,748
	3	0,641	0,787	0,607	0,755
4er-Zelle	1	0,587	0,687	0,604	0,704
	2	0,676	0,834	0,676	0,842
	3	0,681	0,860	0,691	0,871
5er-Zelle	1	0,600	0,700	0,584	0,684
	2	0,681	0,846	0,667	0,829
	3	0,690	0,867	0,679	0,856

Es zeigt sich, dass das Einbringen einer zweiten Kante einen äußerst positiven Effekt auf die Formabweichung der Zelle hat gegenüber einer einzelnen Kante, während eine dritte mögliche Kante nur noch eine geringe Verbesserung der Strukturperformance ermöglicht. Von einer vierten oder fünften Kante wird daher nicht erwartet, dass diese das theoretisch bestmögliche Strukturverhalten signifikant verbessern. Neben dem nicht realistisch handhabbaren benötigten Ressourcenaufwand für $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| = 3$ beim Trainingsdatensatz wird hiermit auch mechanisch gezeigt, dass das Einbringen einer dritten Kante im Durchschnitt keinen deutlichen praktischen Mehrwert generiert, unabhängig von der Bestrafung p_{ins} der Rewardfunktion. Dadurch lässt sich das gewählte Vorgehen weiter motivieren.

5.7.4 Aufbau der datensatzbasierten Umgebung

Die datensatzbasierte Umgebung ist vom Grundkonzept identisch mit der in Abschnitt 5.5 vorgestellten simulationsbasierten Umgebung. Der zentrale Unterschied ist, dass das Generieren von Strukturgraphen, Zellengraphen und Lastfällen wegfällt und stattdessen das bereits berechnete Sample eingeladen wird. Entsprechend ändert sich hauptsächlich nur die Implementierung des Reset- und des Step-Moduls. Diese gliedern sich ansonsten gleichermaßen in den in Abbildung 5–3 vorgestellten Ablauf einer Episode ein. Abbildung 5–28 zeigt das Reset- und Step-Modul für die vorliegende datensatzbasierte Umgebung.



(a) Reset-Modul **(b)** Step-Modul

Abbildung 5–28: Ablaufdiagramme der Module der datensatzbasierten Umgebung

5.7.5 Wahl relevanter Observationsfeatures

Für die Parameterstudie zur Wahl relevanter Features für den Observationsraum auf dem vorberechneten Datensatz werden 256 Agenten in unterschiedlichen Feature-Konfigurationen auf einer Zelle mit $|\mathcal{S}^{\{G_{\mathrm{Zm}}\}}| = 4$ Seiten trainiert. Eine Feature-Konfiguration beschreibt dabei, welche Features aus dem Observationsraum für das Training eines Agenten genutzt werden. Grundlage dafür ist ein DoE, nach dem die zu trainierenden Konfigurationen gewählt werden. Hierfür wird eine LHS-Strategie herangezogen.

Da die Untersuchung einen Einblick in die Relevanz der Features des Observationsraums geben soll, werden andere Hyperparameter des Trainingsalgorithmus und des Agenten bzw. der Strategie nicht variiert. Nichtsdestotrotz müssen diese für das Training gewählt werden, weshalb mit wenigen Ausnahmen die Standardwerte von STABLE-BASELINES3 herangezogen werden, welche in Tabelle 5–8 dargestellt sind. Die Lernrate (vgl. Gleichung (3–15)) beschreibt einen Ausgangswert für die Lernrate, da der Adam-Algorithmus während des Trainings eine individuelle Lernrate für alle Gewichte des NNs bestimmt.

Zu den vom Standardwert abweichenden Parametern gehört

- der Diskontierungsfaktor γ, welcher standardmäßig 0,99 beträgt und hier zu 1,0 gewählt wird, um alle vom Agenten eingesammelten Rewards gleich zu gewichten. Dadurch soll die bestmögliche Strategie über alle einziehbaren Kanten hinweg gefunden werden. Aufgrund der kurzen Episodendauer werden keine numerischen Probleme diesbezüglich erwartet.

Tabelle 5–8: Gewählte PPO-Parameter und Strategieparameter auf Basis der Standardwerte aus STABLE-BASELINES3 (Ausnahmen durch (*) gekennzeichnet) zur Identifikation relevanter Features des Observationsraums

PPO-Parameter	Wert	Strategieparameter	Wert
Optimierungsalgorithmus	Adam	Anzahl Layer des Actors	$1^{(*)}$
Lernrate	$3 \cdot 10^{-4}$	Neuronen je Actor-Layer	64
Batchgröße n_{batches}	64	Anzahl Layer des Critics	$1^{(*)}$
Diskontierungsfaktor γ	$1{,}0^{(*)}$	Neuronen je Critic-Layer	64
Größe des Rollout-Buffers n_{steps}	2048	Aktivierungsfunktion	Tanh
Trainingsepochen n_{epochs}	10	Orthogonale Initialisierung	True
GAE-λ	0,95	CNN-Architektur	Nature-CNN
Strategie-Clipping ϵ	0,2	Einbettungsdimension des CNN	256
Normalisieren der Advantages	True		
Zustandswertkoeffizient c_1	0,5		
Entropiekoeffizient c_2	0,0		
Gradienten-Clipping	0,2		

- Zusätzlich wird die Anzahl der Hidden-Layer des Actors und Critics von 2 auf 1 reduziert. Durch die geringe Komplexität der zugrundeliegenden MLPs soll der Trainingsprozess für den Agenten vereinfacht werden.

In der Dokumentation von STABLE-BASELINES3 werden noch weitere Strategieparameter genannt, welche aber für einen diskreten Aktionsraum nicht genutzt werden und hier entsprechend nicht aufgeführt werden.

Das Training wird im Folgenden basierend auf dem Trainingsdatensatz, den vorgestellten PPO- und Strategieparametern und den 256 Agenten in unterschiedlichen Feature-Kombinationen in der datensatzbasierten Umgebung durchgeführt. Trainiert werden alle Agenten mit individueller Featurekombination für 500 000 Schritte. Für die Ermittlung der Performance der Agenten wird aufgrund der Trainingsdauer eines Agenten von ca. 40 h keine k-Fold-Cross-Validation durchgeführt. Stattdessen wird der Evaluationsdatensatz aus 64 Samples herangezogen.

Zur Evaluierung der Strategie wird eine normierte Variante des Returns $\overline{g_0}_{\gamma=1}$ eingeführt. Stellvertretend wird der normierte Return $\widetilde{\overline{g_0}}_{\gamma=1}$ einfachheitshalber auch Performance des Agenten genannt. Da der maximal erreichbare Return $\overline{g_0}_{\gamma=1,\max}$ des Evaluationsdatensatzes für einen konkreten Zelltypen bekannt ist (vgl. Tabelle 5–7), kann der wahre Return $\overline{g_0}_{\gamma=1}$ entsprechend normiert werden:

$$\widetilde{\overline{g_0}}_{\gamma=1} = \frac{\overline{g_0}_{\gamma=1}}{\overline{g_0}_{\gamma=1,\max}}. \tag{5–21}$$

Ein Wert von 1 ist hier ideal und bedeutet, dass der Agent optimal handelt. Diese Normierung erlaubt es, die Performance von Agenten zwischen verschiedenen Stufen zu vergleichen und die Performance der Agenten besser einzuschätzen.

Der Trainingsverlauf der Agenten mit der zugrundeliegenden Feature-Kombination, die zur besten, schlechtesten und zusätzlich zur Median-Performance führen, wird in Abbildung 5–29 gezeigt. Zusätzlich wird die mittlere Episodenlänge $\overline{L_{\mathrm{Ep}}}$ angegeben, welche grob die Anzahl der in die Zelle eingebrachten Kanten repräsentiert. Hier muss berücksichtigt werden, dass das zweifache Einbringen einer gleichen Kante in die Zelle nur in einer tatsächlichen Kante in der Zelle resultiert, da die zweite Kantenaktivierung ignoriert wird. Nichtsdestotrotz werden dafür zwei Schritte durchgeführt.

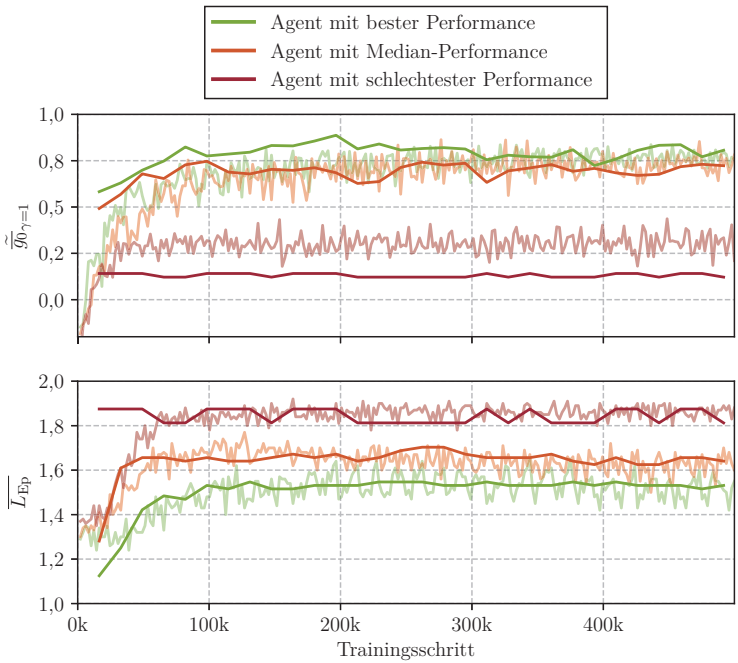


Abbildung 5–29: Trainingshistorien der Performances $\widetilde{\overline{g_0}}_{\gamma=1}$ und Episodenlängen $\overline{L_{\mathrm{Ep}}}$ der Agenten auf dem Evaluationsdatensatz mit den Featurekombinationen des Observationsraums, die zur besten, schlechtesten und zur Median-Performance gehören. Die zugehörigen Rollout-Historien werden in gleicher Farbe semi-transparent dargestellt

Evaluiert werden die Agenten alle 16 384 Schritte. Dabei handelt der Agent stets nach der Aktion, die entsprechend der Wahrscheinlichkeitsverteilung des Actors deterministisch am vielversprechendsten ist. Ebenso wird die Performance der Agenten bei den im

Rollout gesammelten Daten dargestellt. Das sind die Daten aus den Episoden, welche der Agent während des Trainings sammelt. Dabei handelt der Agent stochastisch nach der Wahrscheinlichkeitsverteilung des Actors, weshalb es auch möglich ist, dass der Agent mit einer Wahrscheinlichkeit eine suboptimale Aktion ausführt. Für die Rollout-Daten wird $\widetilde{\overline{g_0}}_{\gamma=1}$ und $\overline{L_{\mathrm{Ep}}}$ über die 100 letzten Episoden gemittelt.

Zu Beginn sind die Gewichte und Bias des Actors und des Critics zufällig initialisiert und die Agenten handeln entsprechend willkürlich. Wie aus den Rollout-Historien erkennbar ist, macht sich das in einer initialen Performance von $\widetilde{\overline{g_0}}_{\gamma=1} \approx -0{,}2$ für alle dargestellten Agenten bemerkbar. Direkt zu Beginn des Trainings, nach den ersten Updates der Strategien, erfahren die Agenten die größte Performance-Änderung. Nach 16 384 Schritten werden die Agenten das erste Mal auf dem Evaluationsdatensatz deterministisch evaluiert. Hier ist die Performance der Agenten durch das vorangegangene Training bereits moderat hoch.

Weiteres Training verbessert die Performance des besten Agenten und des Median-Agenten nur noch geringfügig. Für diese Agenten lässt sich auch erkennen, dass sich die Rollout-Historie, wie zu erwarten, der Evaluierungshistorie annähert. Das liegt daran, dass der Agent mit zunehmendem Training immer sicherer ist, welche Aktionen sinnvoll sind und sich die Wahrscheinlichkeiten der Aktionsverteilung entsprechend anpassen. Entsprechend ungewöhnlich ist das Verhalten des Agenten mit der schlechtesten Performance. Hier ist das stochastische Verhalten des Agenten während des Trainings meist besser als das deterministische Verhalten der Evaluation. Das ist ein signifikantes Indiz dafür, dass der Agent es mit den gegebenen Features für den Observationsraum nicht schafft, sauber zu trainieren.

Weiterhin ist auffällig, dass die Median-Performance und der Agent mit bester Performance recht nah beieinander liegen, was dafür spricht, dass es viele Feature-Kombinationen gibt, welche gute Performances liefern. Mit einer maximalen Performance von $\widetilde{\overline{g_0}}_{\gamma=1} = 0{,}89$ in Trainingsschritt 196 608 des besten Agenten auf den Evaluierungsdaten schafft dieser es, nah an den theoretisch bestmöglichen Wert von 1,0 zu kommen.

Für die Episodenlängen zeichnet sich der Trend ab, dass die Agenten mit besserer Performance im Mittel weniger Kanten in einer Episode in eine Zelle einbringen. Das deckt sich mit der Definition der Rewardfunktion, welche eine Bestrafung für jede eingezogene Kante vorsieht.

Tabelle 5–9 zeigt, welche Features für die zugehörigen Agenten genutzt werden. Zwar ist die Tabelle zur Transparenz und Nachvollziehbarkeit der Trainingshistorien wichtig, doch ist die Aussagekraft begrenzt, da nicht eindeutig daraus hervorgeht, welche Features einen sinnvollen Beitrag zum Training der Agenten beigetragen haben.

Tabelle 5–9: Feature-Konfigurationen der Agenten aus Abbildung 5–29

Feature	Bester Agent	Median-Agent	Schlechtester Agent
Koordinaten	-	-	-
Verschiebungen	✓	-	-
Kantenlängen	✓	✓	-
Wandstärke	✓	✓	-
Gesamtlänge	✓	✓	-
Herstellbarkeit	✓	-	-
Zellenanbindung	-	-	-
Adjazenzvektor	-	-	-
Innere Energiedichte	✓	✓	-
Normalspannungen	-	✓	-
Schubspannungen	✓	-	-
Deformationsbild	-	-	✓

Deshalb wird im Folgenden systematisch analysiert, welche Features besonders zum Training der Agenten geeignet sind. Das geschieht basierend auf dem durchgeführten DoE. Ob ein Feature für den Observationsraum geeignet ist, wird durch die PFI und der Korrelation (vgl. Abschnitt 3.2.3) bzgl. der Performance der Agenten beschrieben.

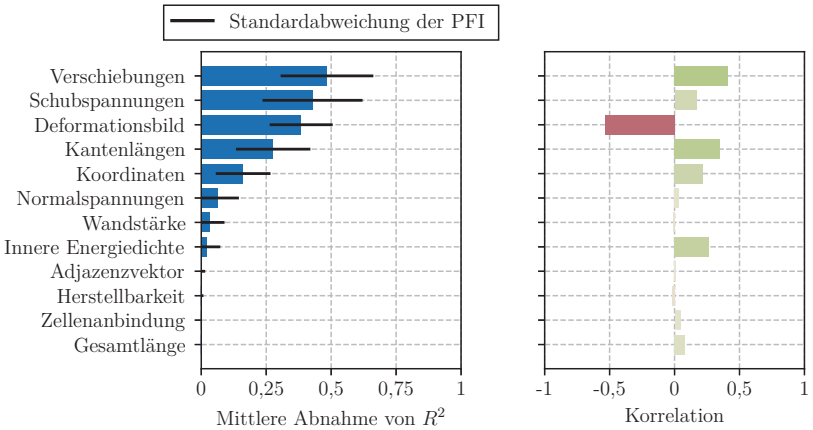
Zur Berechnung der PFIs wird ein RF-Regressionsmodell aus 32 einzelnen DTs in einer k-Fold-Cross-Validation mit $k = 5$ auf den Trainingssplits der insgesamt 256 Agenten trainiert. Die Features beschreiben, ob ein jeweiliges Feature aktiv oder inaktiv ist und als Label bzw. Zielgröße wird die maximale Performance $\widetilde{\overline{g_0}}_{\gamma=1}$ der Agenten auf den Evaluierungsdaten über die Trainingsschritte genutzt.

Die PFIs werden für jedes Feature in Abbildung 5–30 aufgezeigt. Zusätzlich sind die Standardabweichungen der PFIs aufgetragen, welche aus den 32 einzelnen DT-Schätzern resultieren. Das mittlere R^2 der RF-Modelle über alle fünf Folds ist 0,69.

Insgesamt fällt auf, dass einige Features eine signifikante PFI besitzen. Es gibt hingegen nur wenige Features, welche eine moderate bis starke Korrelation zu der Agentenperformance aufweisen.

Eine Auswertung alleinig auf den beiden genannten Kenngrößen ist nicht ausreichend. Ebenso müssen andere Aspekte berücksichtigt werden:

- Die Auswertung gilt so nur für 4er-Zellen und die gewählten Hyperparameter. Andere Zellen und Hyperparametereinstellungen werden hier, um den Aufwand überschaubar zu halten und konsistente Modelle generieren zu können, nicht untersucht. Daher muss ingenieurmäßig abgeschätzt werden, welchen Einfluss die genannten Features auf andere Zellengrößen und Hyperparameter haben könnten.



(a) Permutation Feature Importance (PFI) **(b)** Rangkorrelation

Abbildung 5–30: PFI und Rangkorrelation der Observationsfeatures mit der maximalen Performance $\widetilde{\overline{g_0}}_{\gamma=1}$ der trainierten Agenten

- Manche Features scheinen laut PFI ggfs. als unwichtig bezüglich der Performance der Agenten. Beim Erstellen und Bewerten der Features ist es die Aufgabe als Ingenieur*in, Expertenwissen einfließen zu lassen. Insbesondere vor dem Hintergrund, dass die Performance der Agenten bei anderen Zellengrößen und Hyperparametereinstellungen eventuell besser auf mechanisch einflussreiche Features eingehen kann.
- Es kann sich herausstellen, dass die Verteilung der Werte eines Features im Training nicht der Verteilung in realen Aufgabenstellungen entspricht. Dieses Phänomen kann als *Data Shift* bezeichnet werden. Dabei handelt es sich um ein ähnliches Phänomen wie der bereits besprochene Data Drift, bei dem sich die Verteilung von Features über die Zeit hinweg ändert. Der Data Shift bezieht sich auf abrupte Änderungen zwischen zwei Featureverteilungen. Beispielsweise dann, wenn die Verteilung eines Features beim Training der Agenten anders ist als in der Anwendung der Agenten. Als Resultat muss das Actor-Modell dann Kanten für Observationen vorschlagen, die außerhalb des Trainingsbereichs liegen. Insbesondere bei komplexen Aufgaben leidet die Vorhersagegüte des Modells in einem unbekannten Maß darunter, weshalb es besser sein kann, das Feature nicht zu betrachten. Zwar sollten solche Effekte durch ein entsprechendes Feature-Engineering nicht auftreten, aber je nach Feature lässt sich dies nicht vollständig vermeiden.
- Grundsätzlich ist ein kleiner Observationsraum besser als ein unnötig großer, der nur marginal mehr Informationen bereitstellt. Dadurch kann der Agent die Observationen besser anlernen.

Basierend auf diesen Kriterien wird im Folgenden begründet, welches Feature im Observationsraum genutzt wird und welches nicht. In Tabelle 5–10 wird eine Übersicht über die finale Wahl der Features gegeben.

Tabelle 5–10: Übersicht über die gewählten Features des Observationsraums für alle folgenden Untersuchungen

Feature	PFI	Korrelation	Verwendet	Begründung
Koordinaten	0,18	0,22	✓	Definition der Zellengeometrie
Verschiebungen	0,51	0,41	✓	Hohe PFI und Korrelation
Kantenlängen	0,28	0,35	✓	Moderate PFI und Korrelation
Wandstärke	0,03	−0,01	✓	Mechanische Relevanz
Gesamtlänge	−0,03	0,08	✓	Mechanische Relevanz
Herstellbarkeit	−0,01	−0,02	-	Kein Einfluss
Zellenanbindung	−0,01	0,05	-	Kein Einfluss
Adjazenzvektor	0,00	0,01	-	Kein Einfluss
Innere Energiedichte	0,02	0,27	✓	Moderate Korrelation
Normalspannungen	0,06	0,04	-	Data Shift
Schubspannungen	0,40	0,17	-	Data Shift
Deformationsbild	0,39	−0,54	-	Starke negative Korrelation

Koordinaten: Das Koordinaten-Feature zeigt eine moderate PFI und positive Korrelation zur Agentenperformance. Ingenieurtechnisch betrachtet ist dieses Feature entscheidend für die exakte geometrische Beschreibung der Zelle. Daher wird es als wichtiger Bestandteil des Observationsraums aufgenommen, obwohl es nicht Bestandteil des besten Agenten aus Tabelle 5–9 ist.

Verschiebungen: Das Verschiebungs-Feature beeinflusst mit einer hohen PFI und einer verhältnismäßig hohen Korrelation maßgeblich die Performance der Agenten. Deshalb sind die Verschiebungen eines der zentralen Features für ein erfolgreiches Agententraining. Entsprechend sind sie Bestandteil des Observationsraums.

Kantenlängen: Auch wenn sich die Kantenlängen theoretisch aus dem Koordinaten-Feature ableiten lassen, suggeriert die moderate PFI und positive Korrelation deren eigenständige Einbindung in den Observationsraum. Das ermöglicht dem Agenten eine direkte Einschätzung der Kantenlängen, ohne diese aufwändig aus den Koordinaten bestimmen zu müssen.

Wandstärke: Trotz der sehr geringen PFI und keiner messbaren Korrelation besitzt das Wandstärken-Feature aus mechanischer Sicht eine hohe Relevanz, da sie sowohl das mechanische Verhalten der Struktur beschreibt als auch die Erfüllung der Fertigungsrestriktionen direkt beeinflusst. Die Wandstärke wird unabhängig vom Zelltypen durch einen Skalar repräsentiert, wodurch die Einbindung in den Observationsraum lediglich eine vernachlässigbare Erhöhung der Komplexität mit sich bringt. Aufgrund der niedrigen PFI ist zudem das Risiko einer negativen Auswirkung auf die Agentenperformance gering, was die Aufnahme in den Observationsraum rechtfertigt.

Gesamtlänge: Für dieses Feature lässt sich analog zur Wandstärke argumentieren. Auch dieses Feature wird nur durch einen Skalar beschrieben. Allerdings ist dieses Feature mechanisch besonders wichtig, da es die einzige, leicht zugängliche Möglichkeit für den Agenten ist, abschätzen zu können, wie stark die Wandstärkenänderung beim Einbringen einer neuen Kante ist und ob das Modell dadurch die zugehörige Fertigungsrestriktion verletzt. Entgegen der PFI und Korrelation wird das Feature aufgrund des geringen Risikos einer negativen Auswirkung auf das Training der Agenten und der zugrundeliegenden mechanischen Relevanz in den Observationsraum aufgenommen.

Herstellbarkeit: Das Herstellbarkeits-Feature trägt nicht zum sinnvollen Training eines Agenten bei. Es ist nicht zu erwarten, dass dieses Feature bei unterschiedlichen Zelltypen wichtiger wird. Für den Agenten scheint es weniger relevant zu sein zu erkennen, welche Fertigungsrestriktion verletzt wird. Stattdessen sollen solche Zustände direkt vermieden werden. Für den weiteren Verlauf der Dissertation wird das Feature nicht genutzt.

Zellenanbindung: Obwohl die Zellenanbindung aus mechanischer Sicht bedeutsam ist und dem Agenten Hinweise darauf gibt, wo Wände sich möglicherweise effektiv abstützen können, zeigt sie in der Praxis einen vernachlässigbaren Einfluss auf die Performance der Agenten und weist nur eine geringfügige positive Korrelation zur Agentenperformance auf. Anders als bei der Wandstärke und der Kantenlänge können bei der Zellenanbindung aber auch andere Features, wie beispielsweise die Verschiebungen, teilweise Informationen über die Steifigkeit von der Verbindung des Zellrahmens zur äußeren Struktur bereitstellen. Entsprechend wird das Feature der Zellenanbindung nicht im Observationsraum genutzt.

Adjazenzvektor: Der Adjazenzvektor ist für den Observationsraum nicht geeignet und wird deshalb nicht genutzt. Begründet wird das mit dem äußerst niedrigen PFI und der ebenfalls nicht vorhandenen Korrelation zur Agentenperformance. Grundsätzlich ist der Adjazenzvektor wichtig für den Agenten, um zu identifizieren, welche Kanten in einem Zustand in der Zelle eingebracht sind. Diese Information lässt sich allerdings auch implizit aus anderen Features extrahieren. Deshalb kann dieses Feature vernachlässigt werden.

Innere Energiedichte: Aufgrund der moderaten positiven Korrelation der inneren Energiedichte mit der Performance der Agenten wird diese im Observationsraum genutzt. Die Verteilung der inneren Energiedichte der Zelle gibt Auskunft über Bereiche hoher Energie und niedriger Energie, welche zum Finden einer versteifenden Kante bzw. Wand der Zelle wichtig sind.

Spannungen: Die Normalspannungen korrelieren vernachlässigbar wenig mit der Agentenperformance. Die Schubspannungen leisten einen mäßigen Beitrag zur Verbesserung der Performance der Agenten. Nichtsdestotrotz werden beide Spannungsarten nicht im Observationsraum genutzt. Ursache dafür ist ein Data Shift zwischen den Trainingsdaten und den Daten in der Anwendung des Agenten. Das ist deshalb kritisch, weil der Actor in der praktischen Anwendung dann in einem Bereich genutzt wird, für den er nicht trainiert wurde. Dadurch können die Vorhersagen des Actors stark negativ und unvorhersehbar beeinflusst werden, was möglicherweise in einer schlechteren Performance der Agenten resultiert.

Die statistischen Kenngrößen zur Standardisierung der Features (vgl. Abschnitt 5.6.3) werden für jeden Eintrag des Featurevektors, also für jede Kante e_i aus G_{Zm}, einzeln durch das Training ermittelt. Diese geben Auskunft über die Verteilung des betrachteten Features. Die im Folgenden genannten Kennwerte beziehen sich auf einen Agenten des DoEs, welcher sowohl Normal- als auch Schubspannungen im Observationsraum besitzt. Die Kennwerte sind zwischen verschiedenen Agenten aber nahezu identisch, da diese allein auf der zugrundeliegenden Trainingsumgebung basieren.

Der Mittelwert der Normalspannungen ist für alle Wände nahe $0\,\mathrm{N/mm^2}$. Das ist zu erwarten, da Wände in etwa gleich häufig unter Zug- als auch unter Druckbelastung stehen sollten. Analog lässt sich auch für die Mittelwerte der Schubspannungen argumentieren, welche ebenfalls alle nahe $0\,\mathrm{N/mm^2}$ sind. Diese Eigenschaften der Spannungen sind in der Anwendung des Agenten zu erwarten.

Anders verhält sich das bei den Standardabweichungen der Spannungen. Die maximale Standardabweichung über alle Wände beträgt für die Normalspannungen $36{,}36\,\mathrm{N/mm^2}$ und für die Schubspannungen $9{,}25\,\mathrm{N/mm^2}$. Beide Werte sind verhältnismäßig gering und es ist leicht zu erkennen, dass auch Spannungen weit über die Standardabweichungen hinaus auftreten können. Die aus der Trainingsumgebung abgeleiteten Simulationsmodelle sind folglich nicht geeignet, um die Verteilung von Spannungen auf beliebigen Simulationsmodellen zu repräsentieren.

Deformationsbild: Das Deformationsbild wird aufgrund der ausgeprägten negativen Korrelation mit der Performance des Agenten zusammen mit der hohen PFI nicht als Feature des Observationsraums aufgenommen. Eine mögliche Ursache dafür ist, dass die Komplexität durch die Menge an Pixeln des Bildes nicht mit den verhältnismäßig wenigen Trainingsschritten für den Agenten anzulernen ist.

5.7.6 Hyperparameterstudie

Nachdem der Observationsraum final festgelegt ist, wird in der zweiten Stufe eine Untersuchung bezüglich der Hyperparametereinstellungen des PPO-Algorithmus und der Strategieparameter durchgeführt. Parameter des PPO-Algorithmus umfassen dabei die Parameter, die den Lernprozess des Agenten steuern. Zu den Strategieparametern gehört die konkrete quantitative Definition der Architekturparameter des Actor- und Critic-MLPs, die bisher nur konzeptionell und qualitativ beschrieben wurden. Tabelle 5–11 zeigt die untersuchten Wertebereiche der PPO-Parameter und der Strategieparameter.

Tabelle 5–11: Wertebereiche der PPO-Parameter und Strategieparameter als Grundlage für ein LHS zum Trainieren von 256 Agenten für 3er-, 4er-, und 5er-Zellen. Falls kein Wertebereich angegeben ist, wird der Parameter nicht variiert und der Standardwert aus STABLE-BASELINES3 genutzt

PPO-Parameter	Wertebereich	Standardwert
Optimierungsalgorithmus	/	Adam
Lernrate	$[1 \cdot 10^{-5}; 1 \cdot 10^{-3}]$	$3 \cdot 10^{-4}$
Batchgröße n_{batches}	$[2; 128]$	64
Diskontierungsfaktor γ	$[0{,}0; 1{,}0]$	0,99
Größe des Rollout-Buffers n_{steps}	$[128; 4096]$	2048
Trainingsepochen n_{epochs}	$[5; 25]$	10
GAE-λ	$[0{,}0; 1{,}0]$	0,95
Strategie-Clipping ϵ	$[0{,}05; 0{,}5]$	0,2
Normalisieren der Advantages	True oder False	True
Zustandswertkoeffizient c_1	$[0{,}05; 1{,}0]$	0,5
Entropiekoeffizient c_2	$[0{,}0; 0{,}1]$	0,0
Gradienten-Clipping	$[0{,}1; 3{,}0]$	0,5

Strategieparameter	Wertebereich	Standardwert
Anzahl Layer	$[1; 2]$	2
Neuronen je Layer	$[16; 128]$	64
Aktivierungsfunktion	Tanh oder ReLU	Tanh
Orthogonale Initialisierung	True oder False	True

Die Wahl der besten Parameter wird als stark abhängig bezüglich der Anzahl an Zellseiten $|\mathcal{S}^{\{G_{\text{Zm}}\}}|$ und damit der Länge der Featurevektoren im Observationsraum vermutet. Deshalb wird die Untersuchung der Parameter jeweils für 3er-Zellen, 4er-Zellen und 5er-Zellen durchgeführt. Da die finale Auswahl an Features für den Observationsraum durch keinen Agenten in der vorangegangenen Untersuchung repräsentiert wird, kann mit

der Hyperparameterstudie zusätzlich verifiziert werden, dass mit den zuvor gewählten Features des Observationsraums eine gute Performance der Agenten grundsätzlich möglich ist.

Analog zu der Untersuchung der Features des Observationsraums werden je Zelltyp 256 Agenten trainiert. Ausgewählt werden die Parameter nach einem LHS. Die Datenpunkte nach dem LHS sind für alle Zelltypen identisch. Somit werden bei jedem Zelltyp die gleichen Kombinationen an Hyperparametern untersucht und auf dem Evaluationsdatensatz ohne k-Fold-Cross-Validation validiert. Die Strategieparameter, die die Anzahl der Layer und der Neuronen in diesen Layern steuern, werden für den Actor und den Critic identisch gewählt, um die Komplexität des Systems und die Anzahl an Einstellparametern zu reduzieren.

Die Agenten basierend auf Hyperparametereinstellungen, die zur besten, schlechtesten und Median-Performance geführt haben, werden für 4er-Zellen in Abbildung 5–31 dargestellt. Die Historien von 3er-Zellen und 5er-Zellen werden im Anhang in Abbildung A–1 und Abbildung A–2 gezeigt. Diese zeigen ein ähnliches Trainingsverhalten zu dem der 4er-Zellen, weshalb diese hier nicht separat analysiert werden. Durch die Normierung der Performance sind diese für verschiedene Zelltypen miteinander vergleichbar.

Es zeigt sich, dass für alle Zelltypen ein bester Agent existiert, der eine Performance von $\widetilde{\overline{g_0}}_{\gamma=1} > 0{,}8$ erreicht. Somit wird für alle Zelltypen ein Agent gefunden, der die Steifigkeit der Zellen, bemessen am Return aus den Evaluationsdaten, im Mittel auf mindestens 80 % des theoretisch erreichbaren Returns erhöht.

Analog zu der Untersuchung der Observationsfeatures ist erkennbar, dass zu Beginn des Trainings die Performance stark ansteigt und sich dann ein anhaltendes Plateau ausbildet. Es ist erkennbar, dass die Median-Performance der Agenten für alle Zelltypen ähnlich gut abschneidet wie die beste Performance. Das gibt bereits Auskunft darüber, dass es viele Hyperparametereinstellungen gibt, die eine ähnlich gute Performance erbringen.

Auf die Darstellung der PFIs und Korrelationen wird hier verzichtet. Grund dafür ist, dass es das RF-Modell nicht schafft, aus den Hyperparametereinstellungen eine annähernd plausible Performance eines zugehörigen Agenten abzuschätzen. Entsprechend gering ist das mittlere $R^2 \approx 0{,}38$ des RF-Modells auf den ungesehenen Evaluationssplits einer 5-Fold-Cross-Validation. Auch eine Vergrößerung des RFs durch das Einbringen weiterer DTs führt nur zu weiterem Overfitting, aber nicht zu einer verbesserten Vorhersage der Agentenperformance auf ungesehenen Daten. Entsprechend sind auch die daraus abgeleiteten PFIs statistisch nicht aussagekräftig. Für die Untersuchung der Hyperparametereinstellungen ist das allerdings weniger kritisch als bei den Observationsfeatures, da zuletzt nur der beste Agent je Zelltyp von Interesse ist und nicht das zugrundeliegende mechanische Verhalten zu diskutieren ist.

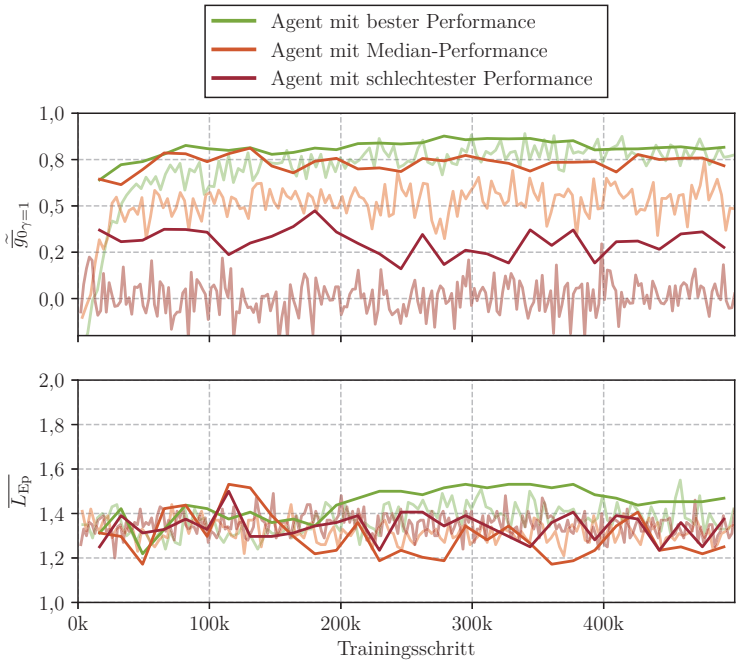


Abbildung 5–31: Trainingshistorien der Performances $\widetilde{\overline{g}}_{0_{\gamma=1}}$ und Episodenlängen $\overline{L_{\mathrm{Ep}}}$ der Agenten auf dem Evaluationsdatensatz von 4er-Zellen mit den Hyperparametereinstellungen, die zur besten, schlechtesten und zur Median-Performance gehören. Die zugehörigen Rollout-Historien werden in gleicher Farbe semi-transparent dargestellt

Nichtsdestotrotz zeigt sich aus einer händischen Analyse der Agentenperformance, dass die Hyperparametereinstellungen der beiden am schlechtesten abschneidenden Agenten zwischen 3er- bis 5er-Zellen identisch sind. Auch andere schlecht abschneidende Hyperparameterkonfigurationen treten für verschiedene Zelltypen auf. Besondere Auffälligkeiten gibt es bei den Parameterkombinationen nicht, sodass hier ein komplexes Zusammenspiel einzelner Parameter als Ursache für das Verhalten vermutet wird. Das wird auch durch die Tatsache gestützt, dass die RF-Modelle Schwierigkeiten haben, den Datensatz zu generalisieren. Analog häufen sich auch die Hyperparameterkonfigurationen der besten Agenten zwischen allen Zelltypen. So sind die besten Hyperparameter für Agenten für 3er- und 5er-Zellen identisch.

Die final gewählten Hyperparameter werden in Tabelle 5–12 für alle drei Zelltypen aufgezeigt. Die Parameter gehören zu den besten Agenten der Hyperparameterstudie aus Abbildung 5–31 (4er-Zelle), Abbildung A–1 (3er-Zelle) und Abbildung A–2 (5er-Zelle).

Tabelle 5–12: Final gewählte PPO-Parameter und Strategieparameter für Agenten auf Basis von 3er- bis 5er-Zellen

PPO-Parameter	3er-Zelle	4er-Zelle	5er-Zelle
Optimierungsalgorithmus	Adam	Adam	Adam
Lernrate	$6{,}663 \cdot 10^{-4}$	$2{,}454 \cdot 10^{-4}$	$6{,}663 \cdot 10^{-4}$
Batchgröße n_{batches}	26	90	26
Diskontierungsfaktor γ	0,225	0,612	0,225
Größe des Rollout-Buffers n_{steps}	3081	3696	3081
Trainingsepochen n_{epochs}	9	9	9
GAE-λ	0,051	0,748	0,051
Strategie-Clipping ϵ	0,261	0,226	0,261
Normalisieren der Advantages	True	True	True
Zustandswertkoeffizient c_1	0,865	0,730	0,865
Entropiekoeffizient c_2	0,018	0,075	0,018
Gradienten-Clipping	1,400	1,578	1,400

Strategieparameter	3er-Zelle	4er-Zelle	5er-Zelle
Anzahl Layer	2	2	2
Neuronen je Layer	30	82	30
Aktivierungsfunktion	ReLU	ReLU	ReLU
Orthogonale Initialisierung	False	False	False

5.7.7 Transferlernen und Modellauswahl

In der dritten Stufe des Trainingsprozesses sollen die in Abschnitt 5.7.6 gefundenen besten Agenten weiter verbessert werden. Dazu werden die trainierten Agenten herangezogen und auf einer leicht modifizierten Aufgabenstellung weiter trainiert. Ein solches Prinzip nennt sich *Transferlernen*. Die neuen Strukturmodelle und Lastfälle und daraus abgeleitete Observationen und Rewards sind bei dieser Untersuchung nicht mehr an einen fixen Datensatz gebunden, sondern werden zufällig während des Trainings entsprechend des Umgebungsaufbaus aus Abschnitt 5.5.1 generiert. Dadurch soll die Abhängigkeit der Agenten von einem Trainingsdatensatz reduziert werden. Ebenso wird das Einziehen

einer dritten Kante in die Zelle durch den Agenten ermöglicht, da nicht alle Kantenkombinationen vorgerechnet werden müssen, wodurch sich die Zellsteifigkeit bei entsprechender mechanischer Ausgangslage theoretisch noch weiter steigern lässt. Ziel ist, dass die Agenten somit komplexere Strukturen generieren können, wenn es das mechanische Verhalten der Zelle erfordert. Zusätzlich soll der Agent durch das Betrachten neuer Observationen eine bessere Verallgemeinerungsfähigkeit ungesehener Daten erlangen.

Durch das Generieren der zufallsbasierten Daten in der konventionellen Trainingsumgebung entsprechend des Aufbaus aus Abschnitt 5.5.1 steigt die zusätzliche Trainingszeit eines Agenten für weitere 200 000 Schritte um ca. 150 h. Aufgrund dieses hohen Zeit- und Ressourceninvests wird das Transferlernen deshalb auch nur für die besten Agenten der Hyperparameteruntersuchung durchgeführt. Eine Zeit von 150 h ist für das Training nur deshalb möglich, da das Sammeln der Trainingsdaten auf dem für die Dissertation zur Verfügung stehenden *High-Performance Computing*-Cluster (HPC-Cluster) stark parallelisiert wird. 32 Trainingsumgebungen werden je Training eines Agenten parallel durchlaufen, wodurch deutlich mehr Daten in gleicher Zeit gesammelt werden. Somit ist der Rollout-Buffer schneller gefüllt und die Strategie des Agenten wird in geringeren zeitlichen Abständen geupdatet.

Die Evaluation der Agenten basiert auf dem gleichen Evaluationsdatensatz bestehend aus 64 Samples wie aus den vorigen Untersuchungen. So wird eine Vergleichbarkeit zwischen den verschiedenen Untersuchungen erzielt. Der Datensatz beinhaltet vorberechnete Kantenkombinationen für bis zu maximal drei eingezogenen Kanten.

Die gesamte Trainingshistorie der Performance $\widetilde{\overline{g_0}}_{\gamma=1}$ von den zugrundeliegenden Agenten aus der Hyperparameteruntersuchung (Stufe 2 mit $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| = 2$) mitsamt Transferlernen (Stufe 3 mit $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| = 3$) wird zusammen mit der durchschnittlichen Episodenlänge $\overline{L_{\mathrm{Ep}}}$ in Abbildung 5–32 aufgezeigt. Normiert ist die Performance $\widetilde{\overline{g_0}}_{\gamma=1}$ durchweg mit dem maximal erreichbaren Return $\overline{g_0}_{\gamma=1,\max}$ für eine maximale Anzahl an einziehbaren Kanten durch den Agenten $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| = 2$. Das erlaubt eine direkte Einschätzung über den Effekt des Transferlernens im Vergleich zu dem zuvor trainierten Agenten. Damit ist es für das Transferlernen zumindest in dieser Abbildung theoretisch möglich ein $\widetilde{\overline{g_0}}_{\gamma=1} > 1$ zu erreichen. Dieser Effekt ist aber gering, da $\overline{g_0}_{\gamma=1,\max}$ unabhängig für $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| = 2$ und $|\mathcal{I}_{\max}^{\{G_{\mathrm{Zm}}\}}| = 3$ sehr ähnlich ist.

Es zeigt sich, dass das Transferlernen keinen positiven Einfluss auf die Performance des Agenten hat. Die maximale Performance der Agenten wird unabhängig von der Anzahl der Zellseiten vor dem Übergang zum Transferlernen (Trainingsschritt $\leq 500\,000$) erreicht. Das ist aber deswegen nachvollziehbar, da die besten Agenten aus der Hyperparameteruntersuchung gewählt wurden und diese einem Bias bezüglich der Evaluationsdaten unterliegen und lokal übermäßig gut performt haben. An diese Performance können die durch das Transferlernen trainierten Agenten nicht anknüpfen. Ebenso ist auch der erwartete Mehrwert durch eine dritte Kante nicht so groß, als dass diese einen deutlichen Mehrwert für den Agenten generieren und den Agent stärker motivieren würde drei

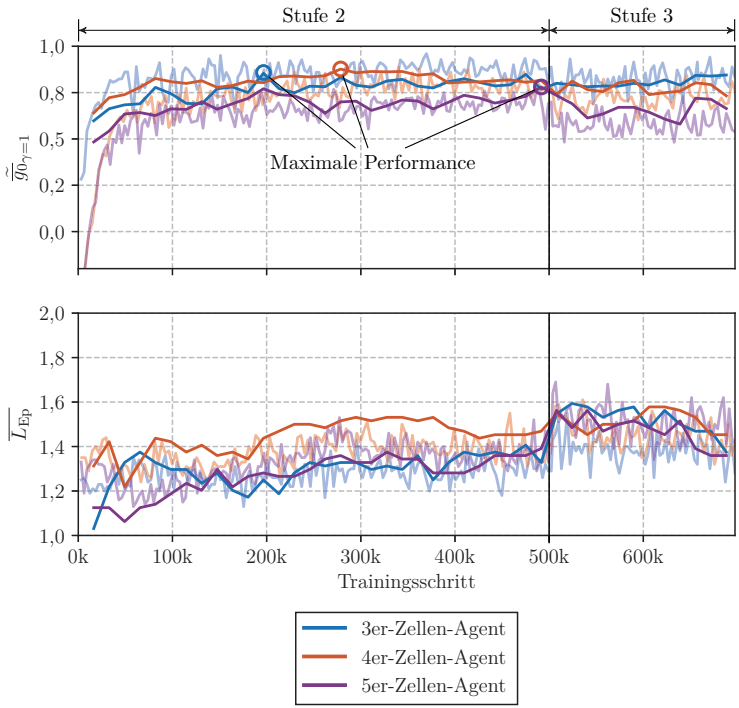


Abbildung 5–32: Trainingshistorien der Performances $\widetilde{\overline{g_0}}_{\gamma=1}$ und Episodenlängen $\overline{L_{\mathrm{Ep}}}$ der besten Agenten aus Stufe 2 (Hyperparameteruntersuchung) für Stufe 3. Die zugehörigen Rollout-Historien werden in gleicher Farbe semi-transparent dargestellt

Kanten einzuziehen. An dem Übergang zum Transferlernen zeigt sich in der mittleren Episodenlänge $\overline{L_{\mathrm{Ep}}}$, dass die Agenten durchaus dazu geneigt sind im Mittel geringfügig mehr Kanten einzuziehen.

Zusätzlich wird in Tabelle 5–13 eine von den Evaluationsdaten unabhängige Schätzung der Agentenperformance auf Basis des bisher ungenutzten Testdatensatzes ermittelt. Die unabhängige Schätzung der Performance wird dort bestimmt, wo $\widetilde{\overline{g_0}}_{\gamma=1}$ über den gesamten Trainingsverlauf auf den Evaluationsdaten maximal ist. Das ist in Abbildung 5–32 markiert. Diese unabhängige Schätzung auf den Testdaten ist repräsentativ für die Agentenperformance. Allerdings darf diese Performance nicht als Grundlage für Modellanpassungen herangezogen werden, da ansonsten die Unabhängigkeit der Schätzung nicht mehr gewährleistet ist.

Tabelle 5–13: Maximale Performance gemessen auf den Evaluationsdaten und zugehörige, repräsentative Performance auf Testdaten für Agenten auf Basis von 3er- bis 5er-Zellen

	Max. $\widetilde{\overline{g_0}}_{\gamma=1}$ auf Evaluationsdaten	Zugehöriges $\widetilde{\overline{g_0}}_{\gamma=1}$ auf Testdaten
3er-Zelle	0,863	0,854
4er-Zelle	0,884	0,844
5er-Zelle	0,791	0,679

Die beste Performance für alle drei Agenten wird in der zweiten Stufe (vgl. Abschnitt 5.7.6) erreicht. Folglich sind die final für die Heuristik gewählten Agenten die Agenten aus der Hyperparameteruntersuchung, welche maximal zwei Kanten in die Zelle einziehen können. Sie zeigen auch auf den Testdaten eine gute bis sehr gute Performance und eignen sich deshalb für die Anwendung in der Heuristik. Nichtsdestotrotz ist der unbeeinflusste Schätzer der Performance bei dem 5er-Zellen-Agent schlechter gegenüber der in der Evaluation ermittelten Maximalperformance, was sich durch die steigende Komplexität des Zellverhaltens bei steigender Seitenanzahl der Zelle erklären lässt.

5.7.8 Anschauliche Bewertung der trainierten Agenten

Bisher wurde die Performance der Agenten rein quantitativ beschrieben. Die anschauliche Auswertung der Agenten aus Abbildung 5–33 zeigt, welche Kanten die Agenten in verschiedenen mechanischen Zuständen der Zellen einbringen und welchen Effekt das auf die Zellsteifigkeit hat, bemessen an der Formabweichung der Zelle.

Das erste Beispiel der Abbildung zeigt eine beliebige 3er-Zelle, welche von dem zugehörigen Agenten verbessert wird. Dadurch, dass der Impaktor direkt auf die Zelle trifft, ist die initiale Zelldeformation groß, was sich durch ein Formabweichungsmaß von $\tilde{A}_\Delta = 0{,}368$ widerspiegelt. Durch das Einziehen der ersten Kanten wird das Formabweichungsmaß auf $\tilde{A}_\Delta = 0{,}269$ und mit der zweiten Kante final auf $\tilde{A}_\Delta = 0{,}166$ reduziert.

Die Reihenfolge, mit der Kanten in den Graphen eingebracht werden, wurde beim Training der Agenten nicht direkt berücksichtigt. Der Diskontierungsfaktor γ bestimmt durch die resultierende Kurz- bzw. Weitsichtigkeit des Agenten indirekt, welche Kanten wann eingezogen werden. Ein praktischer Einfluss des Diskontierungsfaktors γ kann aber basierend auf vorangegangenen Untersuchungen empirisch nicht identifiziert werden, was vermutlich an den kurzen Episodenlängen der Umgebung liegt.

Im Folgenden werden für die 3er-Zelle auch andere Topologien mit unterschiedlich eingebrachten Wänden auf das Formabweichungsmaß hin untersucht, um zu klären, ob eine alternative Herangehensweise signifikant besser abgeschnitten hätte. Hier ist zu beachten, dass die Möglichkeiten zum Einbringen weiterer Wände durch die geringe Anzahl an Rahmenknoten bei 3er-Zellen begrenzt ist.

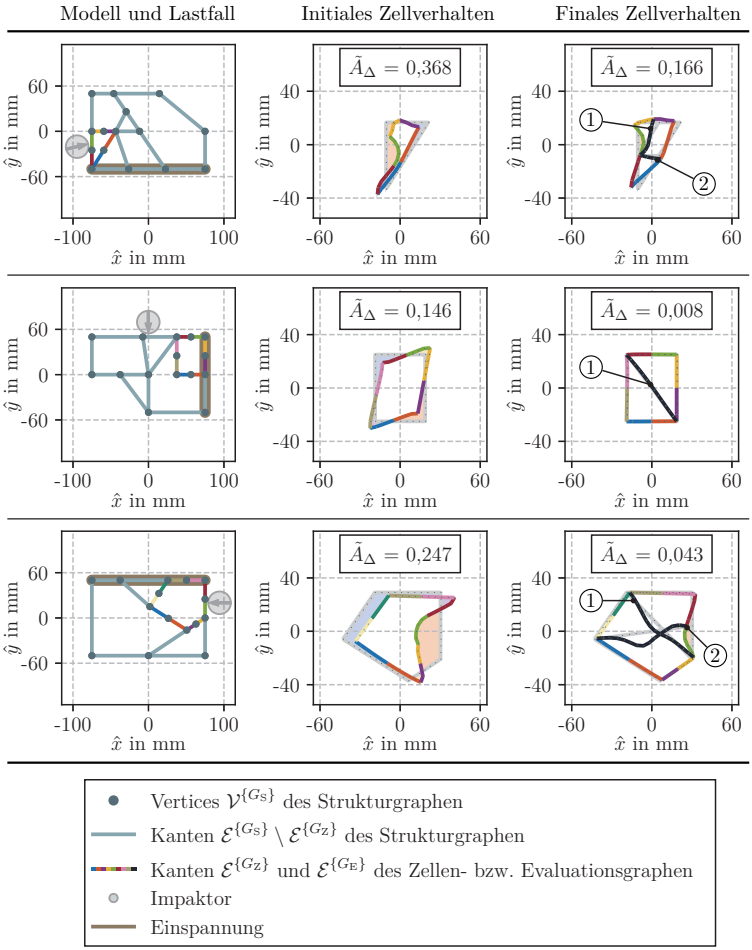
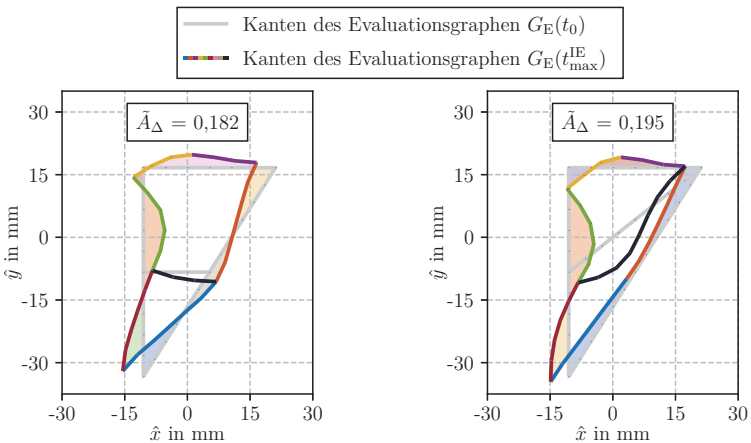


Abbildung 5–33: Anschauliche Performance der finalen Agenten an Beispielmodellen und -lastfällen für 3er- bis 5er-Zellen. Die eingekreisten Ziffern geben die Reihenfolge an, mit der die Kanten eingezogen wurden

Abbildung 5–34a zeigt das Zellverhalten auf, wenn die zweite Kante der 3er-Zelle aus Abbildung 5–33 zuerst eingebracht werden würde. Dann wäre bereits mit einer einzelnen Kante ein Formabweichungsmaß von $\tilde{A}_\Delta = 0{,}182$ erreicht worden. Die beiden eingezogenen Kanten erreichen eine signifikante Versteifung der Zelle. Trotzdem wird die Last durch den Impaktor auf ausschließlich freie Vertices übertragen. Dadurch werden

die angrenzenden Wände eines Segments ineffizient auf Biegung beansprucht. Viele der durch den Kantenteilungsprozess geteilten Kanten werden inhärent nicht durch eine äußere Struktur direkt abgestützt, wodurch das jeweilige Kantensegment — hier sind das das blau-orangene und das lila-gelbe Kantensegment aus dem zugehörigen Strukturgraphen — ineffizient unter einer Biegebeanspruchung belastet wird.

Eine Verbindung zwischen dem Vertex des rot-grünen Segments und dem Eckvertex zwischen der lilanen und orangenen Kante würde es ermöglichen, dass sich die eingebrachte Kante an der Struktur außerhalb der Zelle abstützen könnte. Aufgrund der Länge der entstehenden Wand ist dieser Ansatz aber zu hinterfragen, da sich diese Wand stark unter der eingebrachten Last deformiert. Es wird ein $\tilde{A}_\Delta = 0{,}195$ erreicht. Das zugehörige Deformationsbild wird in Abbildung 5–34b gezeigt.



(a) Zweite eingezogene Kante einzeln

(b) An Eckvertex abgestützte Kante

Abbildung 5–34: Überprüfung alternativer eingezogener Kanten für die 3er-Zelle aus Abbildung 5–33 zur Einschätzung der Performance des Agenten

Im zweiten Beispiel in Abbildung 5–33 wird eine 4er-Zelle analysiert. Durch das Einbringen einer einzelnen Kante durch den zugehörigen Agenten verbessert sich das Formabweichungsmaß von $\tilde{A}_\Delta = 0{,}146$ auf $\tilde{A}_\Delta = 0{,}008$. Mit $\tilde{A}_\Delta = 0{,}008 \leq 0{,}05$ terminiert die Episode automatisch, da die Zelle als steif genug gilt. Dadurch, dass der Impaktor nicht direkt auf die Zelle trifft, erfährt diese einen anderen Belastungszustand als in dem vorigen Beispiel der 3er-Zelle. Die jeweils gegenüberliegenden Vertices an den Zellecken bewegen sich aufeinander zu oder voneinander weg. Dadurch befindet sich die Wand der zugehörigen eingezogenen Kante in einer Druckbeanspruchung. Ingenieurtechnisch wäre die Wahl der anderen Diagonale sinnvoller, da eine Zugbeanspruchung nicht der Gefahr des Knickens bzw. für länger extrudierte Profile des Beulens unterliegt.

Das letzte Beispiel zeigt einen Agenten, der eine 5er-Zelle bezüglich des Formabweichungsmaßes $\tilde{A}_\Delta$ verbessert. Initial hat diese Zelle ein $\tilde{A}_\Delta = 0{,}247$ und erreicht durch das Einbringen der ersten Kante ein $\tilde{A}_\Delta = 0{,}116$ und mit der zweiten und finalen Kante ein $\tilde{A}_\Delta = 0{,}043$. Das Einziehen von lediglich der zweiten Kante macht in diesem Fall aufgrund der großen Kantenlänge nur begrenzt Sinn, da diese knickt. Das wird mit Abbildung 5–35 gezeigt. Damit wird eine Formabweichung von $\tilde{A}_\Delta = 0{,}112$ erreicht. Entsprechend ist es sinnvoll, dass der Agent die knickende Wand zusätzlich abstützt.

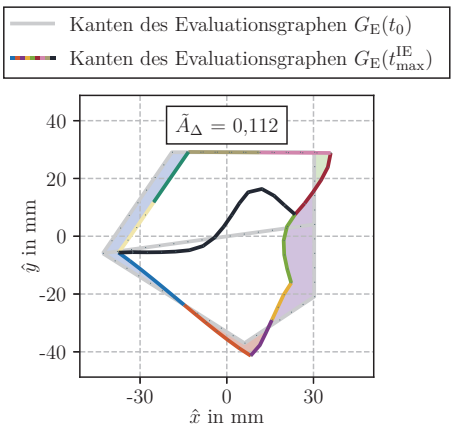


Abbildung 5–35: Überprüfung einer eingezogenen Kante für die 5er-Zelle aus Abbildung 5–33 zur Einschätzung der Performance des Agenten

5.8 Fortführende Diskussion des Trainingsverhaltens

In diesem Abschnitt soll das Trainingsverhalten der Agenten noch einmal tiefergehend untersucht werden. Wie bereits angesprochen wurde, ist der Lerneffekt für die Agenten zu Beginn des Trainings besonders groß. Weiteres Training hat einen vernachlässigbaren Effekt, auch dann, wenn weitere, ungesehene Trainingsdaten dem Training hinzugefügt werden. Das ist verwunderlich, da RL-Algorithmen von Natur aus eine hohe Anzahl an Funktionsaufrufen benötigen und selbst die durchgeführten 500 000 Trainingsschritte im Bereich des RL häufig nicht viel sind. Schlussendlich ist die Anzahl der benötigten Trainingsschritte aber primär von der Komplexität der anzulernenden Umgebung abhängig. Entsprechend kann vermutet werden, dass die Komplexität der Umgebung weitaus geringer ist als ursprünglich angenommen wurde. Typischerweise sind Crashproblemstellungen hochgradig nichtlinear und von einer intrinsischen Komplexität geprägt. Das steht im Gegensatz zu dem scheinbar einfachen Training der Agenten.

Sollte es der Fall sein, dass die Umgebung wirklich leicht für den Agenten anzulernen ist, dann sollte auch eine weitaus vereinfachte NN-Architektur für den Actor und Critic ausreichen, um die Umgebung immer noch gut anzutrainieren. Deshalb wird der beste Agent der Hyperparameterstudie aus Abschnitt 5.7.6 als Referenzagent herangezogen und verglichen mit einem Agenten, bei dem die Hidden-Layer im Actor- und Critic-MLP entfernt wurden. Ansonsten sind die Agenten völlig identisch. So sind die aus der Observation abgeleiteten Inputs direkt mit den Outputs des Actors bzw. Critics über eine Gewichtung verbunden. Das bedeutet, dass beide MLPs jetzt als lineare Approximationsfunktion fungieren und damit die Problemstellung nicht so flexibel anlernen können, wie es im Referenzagent der Fall ist. Da das lineare Actor-MLP lediglich die Logits als Output generiert, werden diese wie gehabt nachträglich in eine multikategorische Verteilung umgewandelt, wobei dieser Prozess nichtlinear ist.

Abbildung 5–36 zeigt einen Vergleich der Performance $\widetilde{\overline{g_0}}_{\gamma=1}$ zwischen dem Referenzagenten und dem Vergleichsagenten ohne Hidden-Layer im Actor- und Critic-MLP.

Abbildung 5–36: Trainingshistorien der Performances $\widetilde{\overline{g_0}}_{\gamma=1}$ und Episodenlängen $\overline{L_{\mathrm{Ep}}}$ des besten Agenten der Hyperparameteruntersuchung für 4er-Zellen und einer modifizierten Variante ohne Hidden Layer im Actor- und Critic-MLP. Die zugehörigen Rollout-Historien werden in gleicher Farbe semi-transparent dargestellt

Es ist kein signifikanter Unterschied der Performance zwischen den beiden Agenten zu erkennen. Zwar ist die Performance des Agenten ohne Hidden-Layer etwas schlechter als die des Referenzagenten, aber die Unterschiede sind geringer als erwartet wurde. Der Effekt kann auch aus dem Bias bezüglich der Evaluationsdaten stammen. Somit schafft es auch eine lineare Approximationsfunktion die Umgebung sinnvoll und effektiv anzulernen, was ein weiteres Indiz dafür ist, dass die Umgebung weniger komplex ist, als zunächst angenommen wurde. Nichtsdestotrotz besteht auch diese lineare Funktion aus $90 \cdot 18 = 1620$ Gewichten und 18 weiteren Bias, wobei die 90 die Anzahl an Input- und 18 die Anzahl an Outputneuronen ist. Die Anzahl an Inputneuronen folgt aus der Anzahl der Einträge des konkatenierten Featurevektors und die Outputneuronen folgen aus $|\mathcal{I}^{\{G_{\mathrm{Zm}}\}}| = 16$ für 4er-Zellen-Agenten mitsamt zwei weiterer Aktionen zur Bestimmung, ob die Episode terminiert werden soll.

5.9 Integration der Heuristik in den Optimierungsprozess

Für die Anwendung der Heuristik wird eine Heuristikumgebung implementiert. In großen Bestandteilen ist der Ablauf ähnlich zur Trainingsumgebung aus Abbildung 5–3, sodass hier die Änderungen zu der Trainingsumgebung beschrieben werden und die Heuristikumgebung nicht von Grund auf neu erklärt wird.

Ein wichtiger Unterschied zur Trainingsumgebung ist, dass im Reset-Modul nun 3er- bis 5er-Zellen gleichzeitig betrachtet werden, denn die Heuristik soll die kritischste Zelle unabhängig der Anzahl an Seiten der Zelle auswählen. Als Priorisierung für den Kantenteilungsprozess wird analog zu der Trainingsumgebung der Flächeninhalt der Zellen genutzt. Ebenso ist ein elementarer Unterschied, dass der Graph und Lastfall bereits existieren und diese nicht generiert werden müssen. Beim Aufruf der Heuristik durch die GHT wird der Pfad zur ASCII-basierten Graph-Syntax des einzuladenden Graphen aus der vorigen Iteration an das Reset-Modul übergeben. Zusätzlich wird der global beliebig orientierte Strukturgraph in ein lokales Profilkoordinatensystem transformiert. Das ist bei der Trainingsumgebung nicht nötig, da die lokale Profilausrichtung der Trainingsgraphen stets mit der lokalen Profilausrichtung übereinstimmt. Die Observationen im Reset-Modul werden in Abhängigkeit des Zelltyps zurückgegeben und zur Verarbeitung für den Agenten mit den in Abschnitt 5.6.3 ermittelten statistischen Kennwerten standardisiert.

Anschließend wird basierend auf der standardisierten Observation eine Aktion vom Agenten gewählt. Im Unterschied zum Training, wo die Aktion aus der Aktionsverteilung gesampelt wird, wird in der Anwendung die vielversprechendste Aktion deterministisch ausgewählt. Diese Aktion wird an das Step-Modul übergeben, wobei das Step-Modul weitestgehend ähnlich ist zu der Trainingsumgebung.

Zuletzt wird der Strukturgraph mit der im Sinne des Formabweichungsmaßes besten Zelle herausgeschrieben und an die GHT zurückgegeben. Üblicherweise ist das die zuletzt generierte Zelltopologie vom Agenten. In Ausnahmefällen, beispielsweise bei einer Fehlentscheidung des Agenten, kann es sich auch um eine frühere Zelle handeln. Dadurch soll die Robustheit des Prozesses gesteigert werden. Die GHT evaluiert dann die Gesamtstruktur bezüglich der originalen Zielfunktionen und Restriktionen. Sie stellt den Agentenvorschlag in Form der sog. RLS-Heuristik in Konkurrenz zu den anderen GHT-Heuristiken aus Abschnitt 2.5.4.

Durch die Art und Weise, wie die Heuristik in die GHT integriert ist, doppeln sich einige Funktionsaufrufe. Das ist ausschließlich der Implementierung der Schnittstelle zwischen GHT und der vorliegenden Heuristik geschuldet. So simuliert die Heuristik im Reset-Modul die Struktur zur Zellevaluation erneut, obwohl die Simulationsdaten aus der vorigen Iteration theoretisch schon vorliegen. Ebenso berechnet die GHT das aus der Heuristik abgeleitete Simulationsmodell erneut, um den Ablauf zur Aktivierung von Heuristiken und die Auswertung der Entwürfe nicht signifikant modifizieren zu müssen.

6 Adaption der entwickelten Heuristik zur Generierung weicher Strukturen

Bisher sollte die Steifigkeit von Zellen auf Basis des Formabweichungsmaßes maximiert werden. In praktischen Optimierungsaufgaben von Crashstrukturen sind allerdings auch Strukturen mit niedrigeren Steifigkeiten, also weicheren bzw. nachgiebigeren Strukturen, relevant. Solche nachgiebigeren Strukturen bauen die kinetische Energie aus dem Crash gegenüber steifen Strukturen über längere Deformationswege ab, wodurch die Verzögerung der Insassen reduziert wird, das Verletzungsrisiko sinkt und die Überlebenswahrscheinlichkeit steigt. Die Aufgabe der Gesamtstruktur ist üblicherweise eine Kompromisslösung aus einer möglichst niedrigen Verzögerung für die Fahrzeuginsassen und einer maximalen Verformung der Struktur, die aus dem verfügbaren Deformationsweg abgeleitet ist. Über die Definition von verschiebungsbasierten Restriktionen und beschleunigungsbasierten Zielfunktionen kann ein solches Verhalten in einer Optimierung beispielsweise sichergestellt werden.

In der GHT sind beschleunigungsbasierte Optimierungsaufgaben häufig von der Dimensionierung in der inneren Optimierungsschleife (vgl. Abbildung 2–14) getrieben. Viele der Heuristiken haben versteifenden Charakter auf die Struktur, welcher durch die Dimensionierung der Entwürfe relativiert wird. Dieser duale Prozess ermöglicht es der GHT eine Kompromisslösung aus geringer Insassenbeschleunigung bei Einhaltung von (Verschiebungs-)Restriktionen zu ermöglichen. Eine Heuristik, die es schafft, die Struktur durch Topologiemodifikationen direkt weicher zu machen, könnte den Optimierungsprozess der GHT weiter unterstützen und die bereits bestehenden Heuristiken sinnvoll ergänzen. Im Folgenden werden deshalb Ansätze vorgestellt, wie die implementierte RL-Umgebung herangezogen werden kann, um mit möglichst wenig Anpassungen an der Umgebung eine weitere Heuristik zu schaffen, mit dem Ziel weichere Zellen und somit weichere Strukturen zu generieren.

6.1 Modifikation des Optimierungsziels des Agenten

Eine naheliegende Herangehensweise zur Adaption der entwickelten Heuristik zur Generierung weicher Strukturen bzw. Zellen ist es, das Optimierungsziel des Agenten anzupassen. Dafür könnten mechanische Größen, wie die von der Struktur absorbierte

Deformationsenergie oder in den Wänden ermittelte Schnittkräfte in einer Zielfunktion verarbeitet werden. So ist die Zelle beispielweise weicher, wenn sie in einem fixen Lastfall eine höhere Energiedichte aufweist als andere Strukturelemente. Eine alternative und einfachere Zielfunktion zur Generierung weicher Zellen ist es, den Agenten darauf zu trainieren, dass dieser das Formabweichungsmaß der Zellen nicht minimieren, sondern maximieren soll. Dazu wird das Vorzeichen der in Gleichung (5–7) definierten relativen Verbesserung $\delta_{t,\mathrm{rel}}$ des Formabweichungsmaßes zur Bemessung der Zellsteifigkeit in der Definition der Rewardfunktion (vgl. Gleichung (5–8)) umgedreht, sodass der Agent eine Belohnung erfährt, wenn das Formabweichungsmaß durch das Einbringen einer neuen Kanten höher wird und somit die Zellsteifigkeit niedriger wird. Die Rewardfunktion ist definiert über

$$r_{t+1} = p_{\mathrm{ins}} + \begin{cases} -\delta_{t,\mathrm{rel}} & \text{falls das Modell fertigbar ist,} \\ -1 & \text{sonst.} \end{cases} \tag{6–1}$$

Grundsätzlich ist es mechanisch schwierig, die Steifigkeit einer Zelle durch das Einbringen von Kanten in die Zelle zu reduzieren. Vereinfacht wird das Vorhaben etwas durch die in der Umgebung eingesetzte Gleichheitsrestriktion bezüglich der Masse. Das Einbringen einer neuen Kante in den Graphen verringert die Wandstärke der Struktur zum Erhalt der initialen Strukturmasse. Nichtsdestotrotz ist die Aufgabe für den anzutrainierenden Agenten signifikant komplexer als das Versteifen der Zelle und der mögliche erreichbare Nutzen bemessen an der negativen relativen Verbesserung geringer. Entsprechend wird für diese Untersuchung die Bestrafung zum Einbringen einer neuen Kante je Schritt zu $p_{\mathrm{ins}} = -0{,}01$ gewählt und ist damit im Betrag deutlich kleiner als bei der Entwicklung der RLS-Heuristik. Da dieser Ansatz aber verhältnismäßig leicht zu testen ist und ggfs. weitere Aufschlüsse über das Trainingsverhalten der Agenten liefert, wird dieser trotzdem weiterverfolgt.

Exemplarisch wird ein Agent für 4er-Zellen trainiert, welcher die gleichen Hyperparametereinstellungen und Features im Observationsraum wie der final gewählte 4er-Zellen-Agent zur Zellversteifung (vgl. Tabelle 5–10 und Tabelle 5–12) hat. Trainiert wird auf dem fixen Trainingsdatensatz für 500 000 Schritte, wobei der Agent bzgl. seiner Performance alle 16 328 Schritte auf dem bereits vorgestellten Evaluationsdatensatz überprüft wird. Da der Datensatz unabhängig von der Optimierungsaufgabe ist, kann dieser direkt verwendet werden. Die benötigten Rewards werden von der datensatzbasierten RL-Umgebung entsprechend der neuen Rewardfunktion basierend auf dem Formabweichungsmaß berechnet.

Die normierte Trainingshistorie auf dem ungesehenen Evaluationsdatensatz wird in Abbildung 6–1 dargestellt. Die Normierung des Returns wird analog zu den vorangegangenen Untersuchungen basierend auf Gleichung (5–21) durchgeführt. Der maximal erreichbare mittlere undiskontierte Return ist für diesen Datensatz $\overline{g}_{0\gamma=1,\max} = 0{,}241$.

Abbildung 6–1: Trainingshistorie der Performance $\widetilde{\overline{g_0}}_{\gamma=1}$ und Episodenlänge $\overline{L_{\mathrm{Ep}}}$ des Agenten auf dem Evaluationsdatensatz von 4er-Zellen zur Generierung nachgiebiger Strukturen. Die zugehörigen Rollout-Historien werden in gleicher Farbe semi-transparent dargestellt

Analog zu den vorigen Untersuchungen ist auch hier die theoretisch bestmöglich erreichbare Performance $\widetilde{\overline{g_0}}_{\gamma=1} = 1{,}0$. Aufgrund der Art und Weise der Normierung ist die Performance nach unten zumindest theoretisch nicht begrenzt. Ein Wert von $\widetilde{\overline{g_0}}_{\gamma=1} \approx 0{,}0$ heißt allerdings, dass der Agent im Mittel über den Evaluationsdatensatz keine signifikante Verbesserung der Nachgiebigkeit der Zelle erreicht. Es fällt auf, dass die mittlere Episodenlänge über das gesamte Training verhältnismäßig niedrig ist. Man würde erwarten, dass das Einbringen von möglichst vielen Kanten bei betragsmäßig geringem $p_{\mathrm{ins}} = -0{,}01$ die Chance die Steifigkeit der Zelle durch Wandstärkenreduktion am besten zu reduzieren. Es zeigt sich allerdings über die gesamte Trainingshistorie hinweg eine globale Tendenz zum Anstieg der mittleren Episodenlänge.

Um eine anschauliche Einschätzung der Agentenperformance zu erlangen, werden in Abbildung 6–2 drei Beispielmodelle und -lastfälle gezeigt, in denen der Agent mit maximalem $\widetilde{\overline{g_0}}_{\gamma=1}$ aus den Evaluationsdaten herangezogen wird und Topologieänderungen vorschlägt. Im ersten und zweiten Beispiel schafft es der Agent das Formabweichungsmaß zu vergrößern und somit die Nachgiebigkeit der Zelle zu vergrößern. Beide Beispiele zeigen dabei einen wesentlichen Mechanismus zur Generierung einer nachgiebigen Zelle.

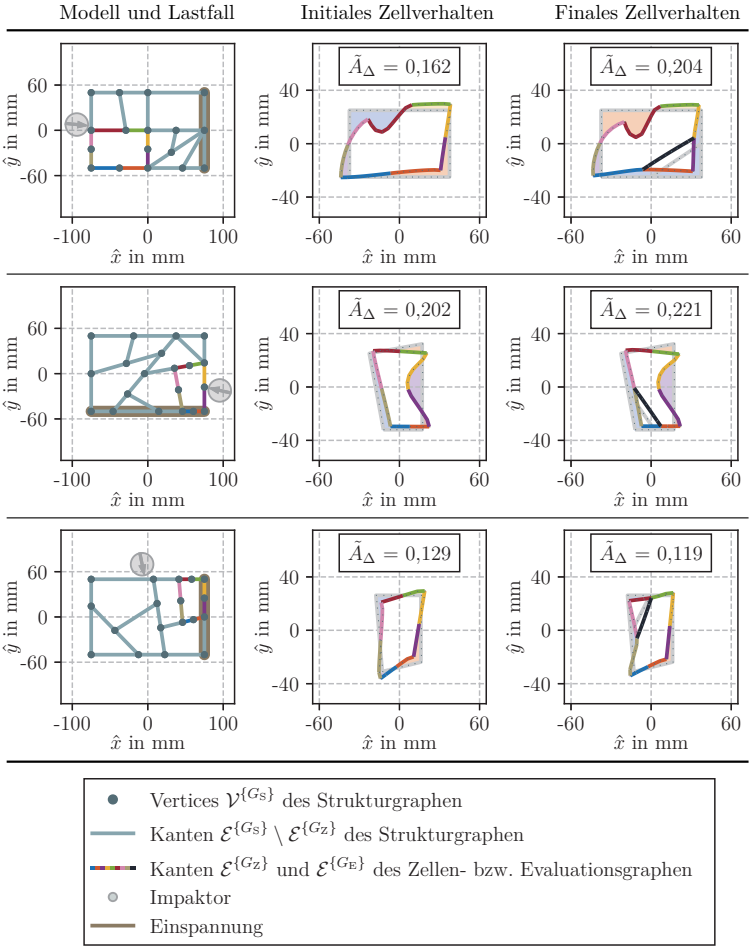


Abbildung 6–2: Anschauliche Performance des Agenten an Beispielsmodellen und -lastfällen für 4er-Zellen zur Generierung weicher Zellen

Der Agent bringt primär dort Kanten ein, wo die Relativverschiebung zwischen den FE-Knoten an den Stellen der Strukturgraphvertices klein ist. So erfüllt die Kante keinen direkten mechanischen Zweck, außer, dass die Wandstärke durch die Massenrestriktion geringer wird. Das dritte Beispiel zeigt, dass dieses Vorgehen aber nicht zwingend in einer weicheren Zelle resultieren muss. Eine Vermutung ist, dass die Vorhersage des Einflusses einer eingebrachten Kante dadurch erschwert wird, dass der Impaktor nicht direkt auf die

Zelle trifft. Dadurch ist die Verformung mehr von der Gesamtdeformation der Struktur abhängig und es treten komplexe Kopplungseffekte auf, welche für den Agenten schwierig einzuschätzen sind.

Der gezeigte Ansatz basiert hauptsächlich darauf, dass die Wandstärke mit der Anzahl an in der Zelle eingezogenen Kanten abnimmt. Grundsätzlich wäre es effektiver, wenn keine neue Kante in die Zelle eingebracht wird, sondern die Wandstärke über den Strukturgraphen direkt reduziert wird. So bleibt die Struktur frei von nahezu willkürlich eingebrachten Kanten, die keinen weiteren gezielten mechanischen Effekt erfüllen. Deshalb wird im Folgenden noch ein alternativer Ansatz vorgestellt.

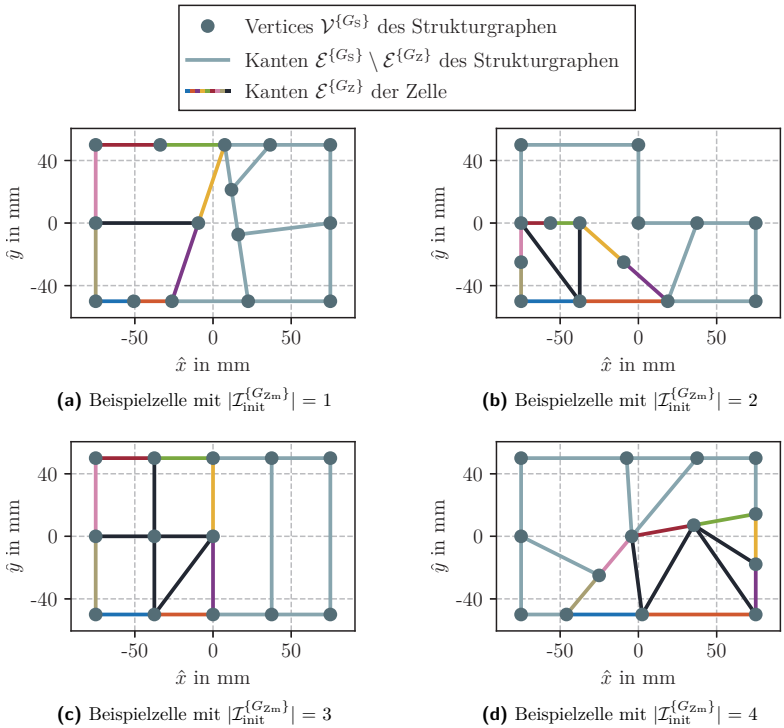
6.2 Entfernen von Wänden aus Zellen

Eine erweiterte Herangehensweise zur Generierung weicher Zellen setzt voraus, dass die Trainingsumgebung tiefergehend angepasst wird. Initial gewählte Zellen sollen hier bereits Kanten beinhalten, welche durch den Agenten iterativ herausgelöscht werden können. Dadurch soll das Generieren weicherer Zellen durch Topologieänderungen ermöglicht werden. Zur Vergleichbarkeit der Zellen ist die Massenrestriktion weiterhin aktiv und erschwert das Training in diesem Fall durch den Anstieg der Wandstärke beim Entfernen von Wänden.

Die Problemstellung ist zum Anlernen eines RL-Agenten komplexer, da potentiell zahlreiche Kanten vom Agenten ausgewählt werden können, die gar nicht in der Zelle existieren. Dieses Problem kann durch eine Anpassung des Rewards adressiert werden, indem der Agent bestraft wird, wenn eine Kante entfernt werden soll, die gar nicht existiert. Eine alternative Herangehensweise ist das sog. *Action Masking* (Huang und Ontañón 2022), bei dem der Agent nur aus einer für den zugrundeliegenden Zustand validen Teilmenge aller Aktionen auswählen darf.

Abbildung 6–3 zeigt verschiedene Beispielzellen in Abhängigkeit der initial bereits eingezogenen Kanten $|\mathcal{I}_{\text{init}}^{\{G_{\text{Zm}}\}}|$ in den modifizierten Zellengraphen G_{Zm}. Analog zu dem Vorgehen aus Kapitel 5 werden Schnittvertices von sich schneidende Kanten innerhalb von Zellen ignoriert, damit die Zellen konsistent beschrieben werden können. Im Beispiel mit $|\mathcal{I}_{\text{init}}^{\{G_{\text{Zm}}\}}| = 3$ tritt dieser Umstand auf. Dort schneiden sich Kanten, wodurch die effektive Anzahl an Kanten in der Zelle G_{Z} höher ist, als mit $|\mathcal{I}_{\text{init}}^{\{G_{\text{Zm}}\}}|$ suggeriert wird.

Damit der Agent sinnvoll entscheiden kann, welche Kanten aus einer Zelle gelöscht werden sollen, ist es wichtig, dass die initiale Zelle bereits mindestens drei Kanten eingezogen hat. Ansonsten ist aufgrund der geringen Auswahl an Kanten eindeutig, welche Kanten gelöscht werden müssen. Dafür würde ansonsten kein RL-Agent benötigt werden. Ebenso benötigt es eines weiteren Kriteriums, damit der Agent nicht die weichste Zelle dadurch



(a) Beispielzelle mit $|\mathcal{I}_{\text{init}}^{\{G_{Zm}\}}| = 1$

(b) Beispielzelle mit $|\mathcal{I}_{\text{init}}^{\{G_{Zm}\}}| = 2$

(c) Beispielzelle mit $|\mathcal{I}_{\text{init}}^{\{G_{Zm}\}}| = 3$

(d) Beispielzelle mit $|\mathcal{I}_{\text{init}}^{\{G_{Zm}\}}| = 4$

Abbildung 6–3: Beispiel von Strukturgraphen G_S mit 4er-Zellen, welche $|\mathcal{I}_{\text{init}}^{\{G_{Zm}\}}| = 1$ bis $|\mathcal{I}_{\text{init}}^{\{G_{Zm}\}}| = 4$ Kanten initial in der Zelle beinhalten

generiert, indem stets alle Kanten aus der Zelle entfernt werden. Hier würde sich z. B. ein mechanisches Kriterium eignen, welches eine minimale Steifigkeit der Zelle fordert. Dafür könnte das Flächenabweichungsmaß herangezogen werden.

In Abhängigkeit der Anzahl der initial eingezogenen Kanten $|\mathcal{I}_{\text{init}}^{\{G_{Zm}\}}|$ in der Zelle wird in Tabelle 6–1 die Häufigkeitsverteilung aller valider Zellen basierend aus den 29 278 zugrundeliegenden Strukturgraphen (vgl. Abschnitt 5.5.3) aufgelistet. Dadurch, dass die zugrundeliegenden Strukturgraphen aus GHT-Optimierungen stammen, ist die ermittelte Häufigkeitsverteilung in guter Näherung repräsentativ für zukünftige GHT-Optimierungen.

Für die 29 278 Strukturgraphen wurden insgesamt 61 877 Zellen mit $|\mathcal{I}_{\text{init}}^{\{G_{Zm}\}}| \geq 1$ gefunden, was bedeutet, dass im Mittel für jeden Graphen ca. 2,1 Zellen gefunden werden, die bereits mindestens eine Kante beinhalten. Somit hätte das Vorgehen grundsätzlich Potential. Es zeigt sich allerdings, dass 98,7 % der Zellen lediglich eine oder zwei Kanten beinhalten.

Tabelle 6–1: Häufigkeitsverteilung der validen Zellen in Abhängigkeit der initial eingezogenen Kanten $|\mathcal{I}_{\text{init}}^{\{G_{\text{Zm}}\}}|$

$\lvert\mathcal{I}_{\text{init}}^{\{G_{\text{Zm}}\}}\rvert$	Häufigkeit	Relative Häufigkeit
1	50 703	$\approx 81{,}9\,\%$
2	10 386	$\approx 16{,}8\,\%$
3	785	$\approx 1{,}3\,\%$
4	3	$\approx 0{,}0\,\%$

Zellen mit größerem $|\mathcal{I}_{\text{init}}^{\{G_{\text{Zm}}\}}|$ treten mit 1,3 % vernachlässigbar selten auf. Dadurch ist das Antrainieren eines Agenten nicht sinnvoll, da die Menge der potentiell löschbaren Kanten klein ist und die Aktionsauswahl zu gering ist.

6.3 Einsatz von etablierten Dimensionierungsstrategien

Bei der Optimierung von weichen Strukturen wird häufig, wie bereits einleitend erwähnt wurde, eine kraft- oder beschleunigungsbasierte Zielfunktion genutzt. Zusätzlich wird i. d. R. eine Restriktion bezüglich der maximal erlaubten Deformation der Struktur festgelegt, welche sich beispielsweise aus dem zur Verfügung stehenden Deformationsraum ergibt.

Zur Generierung weicher Strukturen nutzt die GHT für solche Optimierungsaufgaben Dimensionierungsstrategien, welche die durch die Heuristiken vorgeschlagenen Entwürfe zusätzlich optimieren. Durch die Kombination aus Topologie und Dimension kann das Strukturverhalten exakt auf die gewünschte Optimierungsaufgabe eingestellt werden. Aus diesem Hintergrund heraus wird ersichtlich, dass es in solchen Optimierungsaufgaben auch sinnvoll sein kann eine Heuristik einzusetzen, die die Struktur grundsätzlich versteift.

Entsprechend wird in dieser Dissertation untersucht, wie sich die strukturversteifende RLS-Heuristik in Optimierungsaufgaben zur Kraft- bzw. Beschleunigungsminimierung verhält. Da keine Modifikation an der Umgebung oder dem Agenten vorgenommen werden muss, wird zur Erprobung dieses Ansatzes auf Abschnitt 7.1.3 verwiesen. Dort wird die RLS-Heuristik zur Minimierung der maximalen Beschleunigung des Impaktors an einem Rahmenmodell erprobt.

7 Einsatz der entwickelten Heuristik in praktischen Optimierungen

Bisher wurde die RLS-Heuristik nur innerhalb ihrer Trainingsumgebung evaluiert. In diesem Kapitel wird die RLS-Heuristik in praktischen GHT-Optimierungen eingesetzt. Dazu werden Optimierungen mit aktiver RLS-Heuristik mit denen ohne aktiver RLS-Heuristik verglichen.

Zunächst wird in Abschnitt 7.1 ein Rahmenmodell optimiert. Dabei handelt es sich um das Rahmenmodell, welches auch als Referenz zum Training der Agenten der RLS-Heuristik zum Einsatz gekommen ist und deshalb nur knapp in Abschnitt 7.1.1 vorgestellt wird. Für dieses Modell wird eine Reihe an Untersuchungen durchgeführt. Zum einen wird in Abschnitt 7.1.2 entsprechend der zentralen Aufgabe der RLS-Heuristik eine Optimierungsaufgabe gewählt, welche das Rahmenmodell zunächst durch reine Heuristikaufrufe ohne zusätzliche Formoptimierung versteifen soll. Für diese Optimierungsaufgabe wird anschließend ebenfalls der Einfluss einer in die GHT-Optimierung integrierten Formoptimierung auf die RLS-Heuristik untersucht. Zum anderen wird in Abschnitt 7.1.3 eine Optimierungsaufgabe für das Rahmenmodell ausgewählt, welche die Beschleunigung des Impaktors minimieren soll. Für beide Optimierungsaufgaben wird zusätzlich anhand zweier exemplarischer Konfigurationen an Fertigungsrestriktionen deren Einfluss auf die Funktionsweise der RLS-Heuristik und deren Einfluss auf das gesamte Optimierungsverhalten analysiert.

Neben dem Rahmenmodell wird ein Biegeträger in Abschnitt 7.2 untersucht, welcher mitsamt Lastfall in Abschnitt 7.2.1 vorgestellt wird. In der Optimierung aus Abschnitt 7.2.2 soll die Steifigkeit des Trägers maximiert werden. Der Biegeträger und der zugehörige Lastfall unterscheiden sich in vielen Aspekten signifikant von den Optimierungsaufgaben des Rahmenmodells, wodurch überprüft werden kann, ob die RLS-Heuristik auch für unbekannte Modelle und Lastfälle sinnvolle Vorschläge zur Strukturversteifung generieren kann.

In Abschnitt 7.3 wird ein Schwellerausschnitt optimiert. Dieser wird in Abschnitt 7.3.1 eingeführt und in Abschnitt 7.3.2 bezüglich seiner Steifigkeit optimiert. Um die Wirksamkeit der auf Steifigkeitssteigerung trainierten RLS-Heuristik zu testen, wird anders als bei dem Lastfall üblich, die Eindringung eines Pfahls minimiert. Analog zum Biegeträger ist auch dieses Modell für den Agenten unbekannt. Entsprechend soll auch hier die RLS-

Heuristik in einer Optimierung mit ihr beim Training ungesehenen Modellverhalten erprobt werden. Analog zu den vorigen Untersuchungen werden zwei unterschiedliche Parametersätze an Fertigungsrestriktionen überprüft.

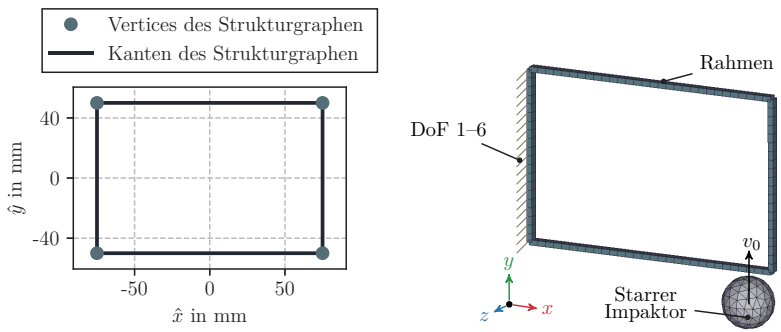
Abschließend wird in Abschnitt 7.4 der Nutzen der RLS-Heuristik in den praktischen Optimierungen diskutiert. Dabei wird der Fokus auf den Einfluss der Heuristik auf die Zielfunktion, auf das Optimierungsverhalten und die Anzahl an benötigten Funktionsaufrufe gelegt.

Für alle Optimierungskonfigurationen dieses Kapitels gilt, dass die GHT maximal 20 Iterationen durchführen darf. Eine frühere Terminierung der GHT ist dann möglich, wenn nach zwei Iterationen keine Verbesserung der Zielfunktion erzielt werden kann. Es werden die fünf besten Entwürfe an die nächste Iteration übergeben.

7.1 Rahmenmodell

7.1.1 Modellvorstellung

Das Rahmenmodell wurde bereits in Abschnitt 5.4 detailliert vorgestellt. Alle dort definierten Modell- und Lastfallparameter werden für diese Untersuchung übernommen. Zur Vollständigkeit wird dieses Modell zumindest visuell noch einmal in Abbildung 7–1 dargestellt, damit relevante Raumrichtungen für die Optimierungsaufgabe klar definiert sind. Die lokale $\hat{y}$-Richtung des Profils entspricht hier der globalen y-Richtung. Die linke Kante des Strukturgraphen ist im FE-Modell fest eingespannt, d. h., dass alle translatorischen und rotatorischen DoF gesperrt sind.



(a) Strukturgraph des Rahmens in Profilkoordinaten

(b) FE-Modell des Rahmens inklusive Lastfall

Abbildung 7–1: Strukturgraph des Rahmens, welcher mittels GRAMB in das zugehörige FE-Modell übersetzt wird

Der Grund für die Wahl dieses Modells zur Optimierung ist, dass der Agent der RLS-Heuristik auf vergleichbaren Modellen und Lastfällen trainiert wurde. Entsprechend ist zu erwarten, dass diese Heuristik in einer solchen Optimierung das mechanische Verhalten des Modells gut einschätzen kann und sinnvolle Beiträge zur Optimierung beisteuert. So kann zusätzlich ein besonderer Fokus darauf gelegt werden, welchen Effekt die Betrachtung lokaler Zellen im Vergleich zur eigentlichen Optimierung der globalen Zielfunktion hat.

7.1.2 Minimierung der maximalen Impaktorverschiebung

In diesem Optimierungsbeispiel soll die Steifigkeit des Rahmenmodells maximiert werden. Bemessen wird das an der maximalen Verschiebung des Impaktors in y-Richtung $u_{y,\max}$, welche folglich minimiert werden soll. Zur Überprüfung der Herstellbarkeit werden die Kantenlängen $l_{e_i}^{\{G_\mathrm{S}\}}$, die Kantenabstände $d_{e_i,e_j}^{\{G_\mathrm{S}\}}$, die Winkel zwischen Kanten $\alpha_{e_i,e_j}^{\{G_\mathrm{S}\}}$ und die Blechdicke $t_{e_i}^{\{G_\mathrm{S}\}}$ des Strukturgraphen G_S restringiert. Zusätzlich soll die initiale Masse des Rahmens m_init durch Skalierung der Wandstärken konstant gehalten werden. Im Folgenden werden Optimierungskonfigurationen mit

- lockeren Fertigungsrestriktionen,
- praxisnahen Fertigungsrestriktionen und
- einer Optimierungsaufgabe mit Formoptimierung

vorgestellt. Durchgeführt wird jede dieser Optimierungskonfigurationen mit je einer Optimierung mit und ohne aktiver RLS-Heuristik, um den Effekt der Heuristik auf die Optimierung besser bewerten zu können.

Für die ersten beiden Optimierungskonfigurationen gilt, dass die Form und Topologie der Entwürfe ausschließlich durch die Heuristikaufrufe bestimmt wird. Die GHT führt dann keine Formoptimierung oder Dimensionierung durch.

Lockere Fertigungsrestriktionen Die möglichen Querschnittsgeometrien der durch Strangpressen hergestellten Profile sind fertigungstechnisch limitiert. Um den Einfluss der Fertigungsrestriktionen auf die optimierten Strukturen zu beschreiben und die Optimierungsaufgabe für die RLS-Heuristik zu vereinfachen, werden zunächst lockere Fertigungsrestriktionen herangezogen, welche in der Praxis in nicht bzw. nur in schwierig durch Strangpressen herstellbaren Profilen resultieren. Beispielsweise können durch die lockeren Fertigungsrestriktionen kleine Kammern entstehen, welche mit diesem Fertigungsverfahren nicht hergestellt werden können.

Die konkrete Optimierungsaufgabe für locker gewählte Fertigungsrestriktionen lautet

$$
\begin{aligned}
\min \quad & u_{y,\max} := \max\, u_y(t), \\
\text{so dass} \quad & l_{e_i}^{\{G_S\}} \geq 5\,\text{mm}, \\
& d_{e_i,e_j}^{\{G_S\}} \geq 5\,\text{mm}, \\
& \alpha_{e_i,e_j}^{\{G_S\}} \geq 15^\circ, \\
& 1\,\text{mm} \leq t_{e_i}^{\{G_S\}} \leq 5\,\text{mm} \text{ und} \\
& m = m_{\text{init}} = 20{,}25\,\text{g}.
\end{aligned}
$$

Die Optimierungshistorie wird anhand des Entwurfsbaums in Abbildung 7–2 gezeigt. Es sind alle Entwürfe dargestellt, die zur Generierung der optimierten Struktur direkt beteiligt sind. Hier werden also nur die Entwurfsderivate gezeigt, die mit der optimierten Struktur eng verwandt sind.

Insgesamt hat die GHT neun Iterationen gebraucht, um den Rahmen bezüglich der y-Verschiebung des Impaktors zu optimieren. Davon sind im Entwurfsbaum nur sechs Iterationen dargestellt, da nach der sechsten Iteration keine Verbesserung der Impaktorverschiebung $u_{y,\max}$ erzielt wird. In der Optimierung wird der Wert der Zielfunktion von 72,36 mm auf 7,12 mm verbessert.

Die RLS-Heuristik hat durch den Topologievorschlag in der ersten Iteration einen entscheidenden Beitrag dazu geleistet. Keine andere Heuristik hat in der ersten Iteration eine diagonale Wand in den Rahmen eingebracht. Gerade diese ermöglicht es der DNW-Heuristik in der dritten und vierten Iteration ein Dreieckprofil zu generieren, welches naturgemäß durch die Form eine hohe Steifigkeit aufweist. Zusätzlich haben die Wandstärken durch das Löschen von Wänden zugenommen, was die Steifigkeit des Rahmens noch weiter erhöht. Nach der vierten Iteration beträgt die y-Verschiebung bereits nur noch 8,08 mm. Die RLS-Heuristik hat zwei Wände in die Struktur eingebracht. Das ist mit den konventionellen Heuristiken nicht möglich, welche je Aufruf lediglich eine Wand in die Struktur einziehen können.

In der fünften Iteration wird eine Topologiemodifikation an dem Auftreffpunkt des Impaktors auf den Rahmen vorgenommen, um dort lokal die Struktursteifigkeit zu erhöhen und eine dortige Deformation und damit eine Vergrößerung der Verschiebung in y-Richtung des Impaktors zu vermeiden. Da die globale Steifigkeit des Rahmens durch lokale Topologiemodifikationen durch die Wandstärkenabnahme geringer wird, wird in der sechsten Iteration eine zusätzliche globale Versteifung zwischen dem Niedrigenergiebereich der Einspannung und dem Hochenergiebereich an dem Auftreffpunkt des Impaktors durch die Heuristik BED eingebracht.

Bei der Angabe von Funktionsaufrufen in Optimierungen mit aktiver RLS-Heuristik ist zu beachten, dass die aktuelle Implementierung und Integration der Heuristik in die GHT effizienter gestaltet werden kann, da manche Funktionsaufrufe von der RLS-

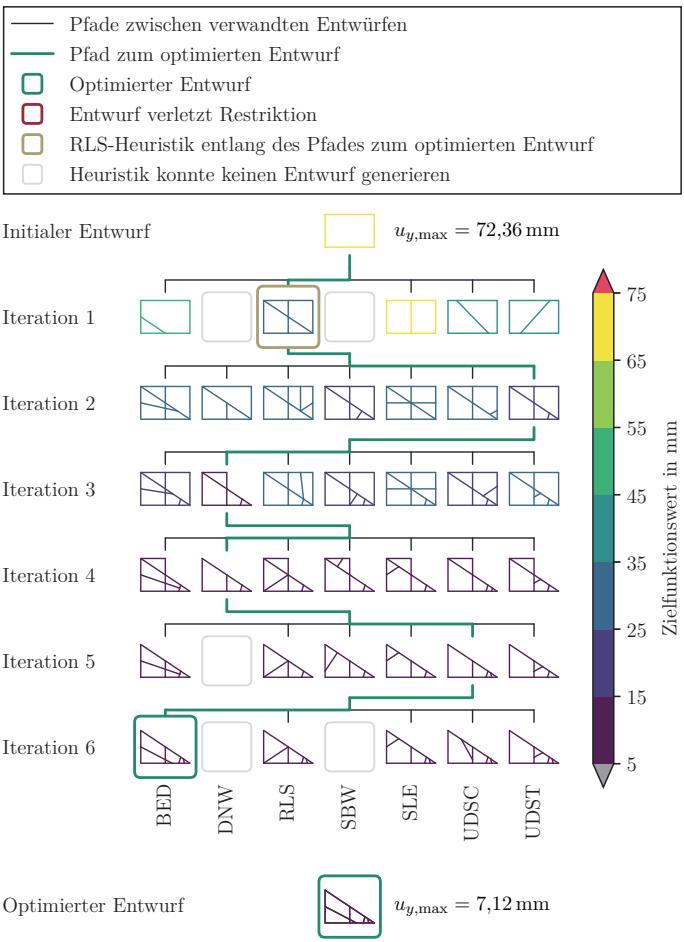


Abbildung 7–2: Entwurfsbaum der Optimierung der Impaktorverschiebung bei dem Rahmenmodell mit *aktiver* RLS-Heuristik unter lockeren Fertigungsrestriktionen

Heuristik und von der GHT unnötigerweise doppelt ausgeführt werden. Das ist auf die aktuelle Implementierung der Schnittstelle zwischen der extern an die GHT gekoppelte RLS-Heuristik und dem Ablauf der GHT zurückzuführen. Deshalb werden bei solchen Optimierungen stets zwei Zahlen zur Beschreibung der Anzahl der Funktionsaufrufe angegeben. Die erste Zahl beschreibt die tatsächliche Anzahl an durchgeführten Simulationen und die zweite Zahl in Klammern beschreibt die Anzahl, die theoretisch bei verbesserter Implementierung der Schnittstelle erreichbar wäre. In dieser Optimierung wurden insgesamt 278 (211) Funktionsaufrufe benötigt.

Zur weiteren Nachvollziehbarkeit der Optimierungshistorie sind die Ergebnisse aller in dieser Dissertation durchgeführten Optimierungen für die einzelnen Iterationen tabellarisch zusammengefasst. Diese finden sich im Anhang wieder, um den Lesefluss nicht zu stören. Die vorliegende Optimierung wird in Tabelle B–1 zusammengefasst.

Als Referenzoptimierung wird die gleiche Optimierung erneut durchgeführt, mit dem Unterschied, dass die RLS-Heuristik inaktiv ist. Der zugehörige Entwurfsbaum wird in Abbildung 7–3 dargestellt. Diese Referenzoptimierung benötigt insgesamt 16 Iterationen mit 336 Funktionsaufrufen, wobei nach 13 Iterationen keine weitere Strukturverbesserung erzielt wird. Der Zielfunktionswert verbessert sich von 72,36 mm auf 11,8 mm. Die Zusammenfassung der Optimierungsergebnisse befindet sich im Anhang in Tabelle B–2. Aufgrund der großen Iterationsanzahl ist der Entwurfsbaum zwischen der fünften und zwölften Iteration gekürzt dargestellt. Durch die Komplexität der Entwürfe ist eine detailreich Analyse dieser Iterationen ohnehin schwierig und nicht zielführend. Stattdessen wird der Fokus auf das globale Verhalten der Optimierung gelegt.

Es fällt auf, dass sich die GHT in der ersten Iteration für einen Entwurf entschieden hat, der für die GHT langfristig sinnvoll ist. Aus ingenieurmäßiger Betrachtung ist die von der Heuristik BED eingezogene Wand fragwürdig, was sich auch in der geringen Verbesserung des Zielfunktionswertes $u_{y,\max}$ widerspiegelt. Der Entwurf der BED-Heuristik besitzt einen Zielfunktionswert von $u_{y,\max} = 54{,}79\,\text{mm}$, während mit der UDSC-Heuristik ein Wert von $u_{y,\max} = 35{,}38\,\text{mm}$ möglich wäre.

Mit der Entscheidung für die UDST-Heuristik in der zweiten Iteration ist die grobe Querschnittsgeometrie festgelegt, welche in den folgenden Iterationen durch viele kurze Wände weiter versteift wird. Von dieser ingenieurmäßig suboptimal eingeschätzten Grundgeometrie schafft es die GHT in den folgenden Iterationen nicht abzuweichen. Stattdessen wird über die folgenden Iterationen eine feinmaschiger Querschnitt generiert, welcher insbesondere im Bereich des Auftreffpunkts des Impaktors ausgeprägt ist.

Wichtig ist zu erwähnen, dass die komplexen Strukturen der Optimierung das Resultat einer fehlenden initialen Formoptimierung sind, welche den gesamten Optimierungsprozess der GHT zu sinnvolleren Entwürfen gelenkt hätte. Diese Formoptimierung wird häufig von dem/der Anwender*in der GHT aktiviert, wenn es die verfügbaren Ressourcen zulassen. In dieser Untersuchung wird auf die Formoptimierung verzichtet, um den Effekt der RLS-Heuristik besser herausstellen zu können.

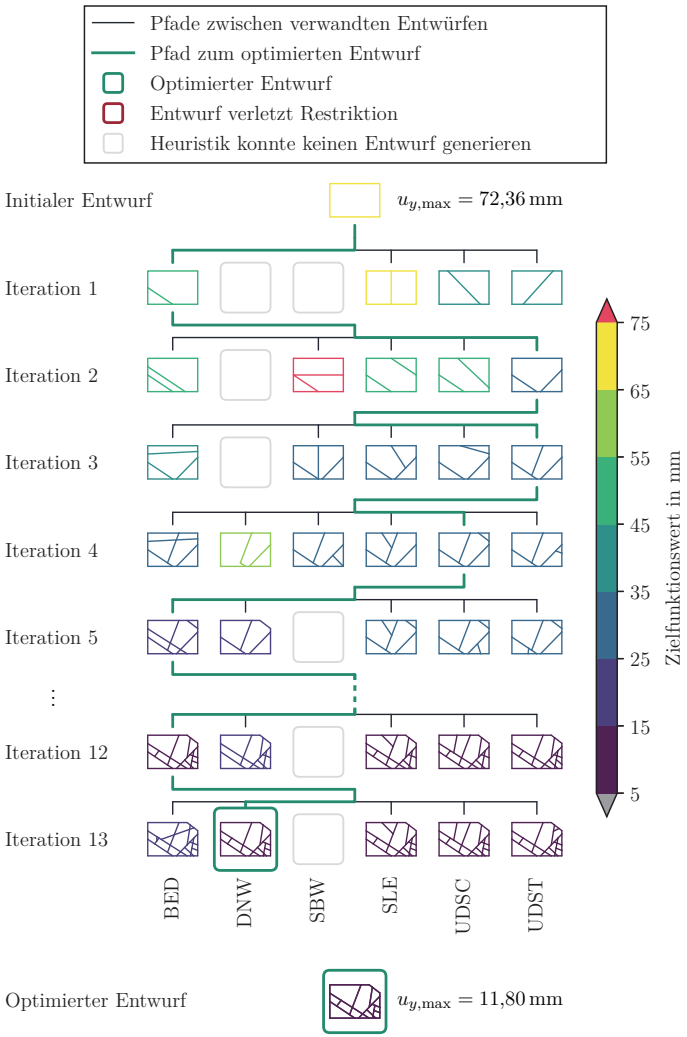
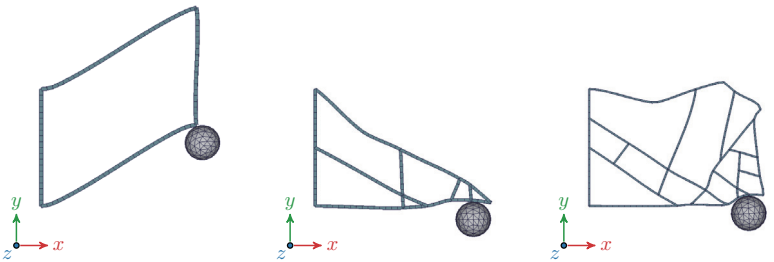


Abbildung 7–3: Entwurfsbaum der Optimierung der Impaktorverschiebung bei dem Rahmenmodell mit *inaktiver* RLS-Heuristik unter lockeren Fertigungsrestriktionen

Abschließend wird in Abbildung 7–4 die Deformation beider optimierten Rahmenprofile dem Startentwurf gegenübergestellt. Während sich die Struktur aus der Optimierung mit aktiver RLS-Heuristik nur geringfügig deformiert und ein nachvollziehbares mechanisches Verhalten aufweist, ist das Deformationsverhalten der Struktur aus der Optimierung mit inaktiver RLS-Heuristik hingegen deutlich komplexer.



(a) Startentwurf $(\cdot)$
$u_{y,\max} = 72{,}36\,\mathrm{mm}$
$t = 13{,}65\,\mathrm{ms}$
$t_{e_i}^{\{G_S\}} = 3\,\mathrm{mm}$

(b) Optimierter Rahmen (RLS aktiv)
$u_{y,\max} = 7{,}12\,\mathrm{mm}$
$t = 1{,}15\,\mathrm{ms}$
$t_{e_i}^{\{G_S\}} = 2{,}41\,\mathrm{mm}$

(c) Optimierter Rahmen (RLS inaktiv)
$u_{y,\max} = 11{,}8\,\mathrm{mm}$
$t = 2{,}15\,\mathrm{ms}$
$t_{e_i}^{\{G_S\}} = 1{,}36\,\mathrm{mm}$

Abbildung 7–4: Deformationsbilder der initialen und optimierten Rahmenstrukturen unter lockeren Fertigungsrestriktionen mit repräsentativer Wanddicke zum Zeitpunkt der minimalen kinetischen Energie des Impaktors

Praxisnahe Fertigungsrestriktionen Die Optimierungsaufgabe lautet analog zu der vorigen Untersuchung

$$
\begin{aligned}
\min \quad & u_{y,\max} := \max\, u_y(t), \\
\text{so dass} \quad & l_{e_i}^{\{G_S\}} \geq 15\,\mathrm{mm}, \\
& d_{e_i,e_j}^{\{G_S\}} \geq 15\,\mathrm{mm}, \\
& \alpha_{e_i,e_j}^{\{G_S\}} \geq 15^\circ, \\
& 1\,\mathrm{mm} \leq t_{e_i}^{\{G_S\}} \leq 5\,\mathrm{mm} \text{ und} \\
& m = m_{\mathrm{init}} = 20{,}25\,\mathrm{g}.
\end{aligned}
$$

Der Entwurfsbaum der zugehörigen Optimierung mit aktiver RLS-Heuristik und praxisnahen Fertigungsrestriktionen wird in Abbildung 7–5 visualisiert. Die GHT hat insgesamt sieben Iterationen mit 197 (151) Funktionsaufrufen berechnet. Davon sind 69 (23) Funktionsaufrufe auf die RLS-Heuristik zurückzuführen. Die optimierte Struktur entstammt der vierten Iteration. In Tabelle B–3 im Anhang sind die Optimierungsergebnisse für die jeweiligen Iterationen dokumentiert.

Durch die Optimierung verbessert sich der Zielfunktionswert $u_{y,\max}$ von 72,36 mm auf 7,82 mm und ist damit nur geringfügig schlechter als die analoge, zuvor gezeigte Optimierung mit aktiver RLS-Heuristik und lockeren Fertigungsrestriktionen bei 81 (60) weniger Funktionsaufrufen.

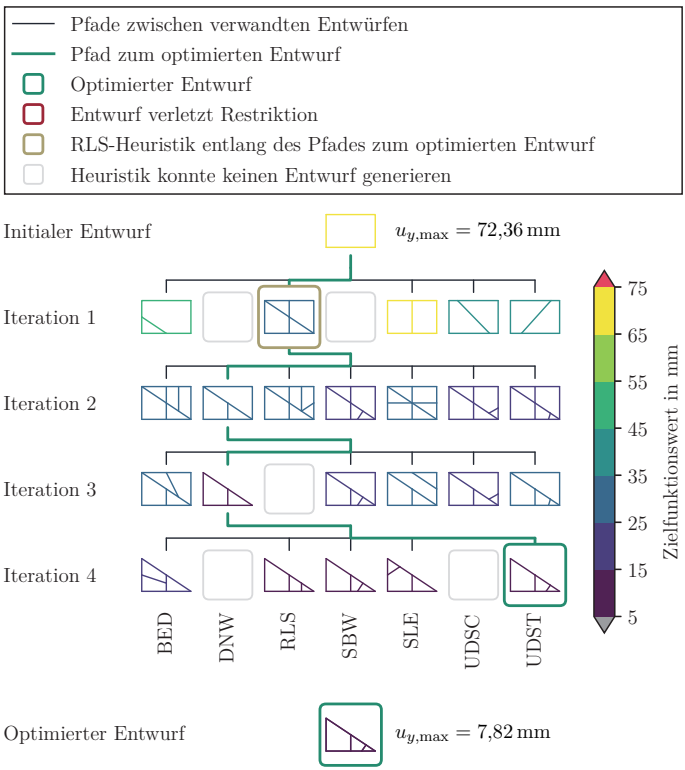


Abbildung 7–5: Entwurfsbaum der Optimierung der Impaktorverschiebung bei dem Rahmenmodell mit *aktiver* RLS-Heuristik unter praxisnahen Fertigungsrestriktionen

Der prinzipielle Ablauf, dem die Optimierung folgt, deckt sich mit dem der Optimierung mit lockeren Fertigungsrestriktionen und aktiver RLS-Heuristik. Der zentrale Unterschied ist, dass aufgrund der strengeren Fertigungsrestriktionen keine lokalen und feinmaschigen Topologiemodifikationen im Bereich des Auftreffpunkts des Impaktors durchgeführt werden und die Struktur dadurch einfacher aufgebaut ist.

Wieder entsteht ein Dreieckprofil, was auf die Kombination der RLS-Heuristik in der ersten Iteration und den DNW-Aufrufen in der zweiten und dritten Iteration zurück geht. Die meisten Entwürfe aus der vierten Iteration sind ähnlich zu dem optimierten Entwurf. So ähnelt der optimierte Entwurf beispielsweise der Form der Entwürfe aus den Heuristiken RLS und SBW. Mit einem Zielfunktionswert von 8,24 mm für den Entwurf der RLS-Heuristik und 8,58 mm für den Entwurf der SBW-Heuristik aus der vierten Iteration sind diese leicht schlechter als die optimierte Struktur der UDST-Heuristik mit 7,82 mm.

Auch für diese Optimierung wird eine Referenzoptimierung durchgeführt, bei der die RLS-Heuristik inaktiv ist. Die zugehörige Optimierungshistorie wird an dem in Abbildung 7–6 dargestellten, gekürzten Entwurfsbaum gezeigt. Insgesamt hat die GHT zwölf Iterationen mit 252 Funktionsaufrufen berechnet, wobei die optimierte Struktur in der neunten Iteration gefunden wird. Dabei wird die Zielfunktion von 72,36 mm auf 14,28 mm verbessert. Eine Zusammenfassung der Optimierung findet sich im Anhang in Tabelle B–4. Auch diese Optimierung weist große Ähnlichkeiten zu der analogen Optimierung mit lockeren Fertigungsrestriktionen auf. Das grundlegende Konzept der Struktur ist vergleichbar, wobei die Kammern entsprechend der Fertigungsrestriktionen etwas größer ausfallen.

Nichtsdestotrotz ist der optimierte Entwurf durch seine Komplexität geprägt, welche auf die fehlende initiale Formoptimierung zurückzuführen ist. Insbesondere die Entscheidung für die Heuristiken BED und UDST in den ersten beiden Iterationen lenkt die GHT-Optimierung in eine ungünstige Richtung. Aufgrund der geänderten Fertigungsrestriktionen sieht der aus den genannten Heuristiken entspringende Entwurf leicht anders aus als in der Optimierung mit inaktiver RLS-Heuristik mit lockeren Fertigungsrestriktionen.

Theoretisch wäre die Aktivierung der UDSC-Heuristik in der ersten Iteration aus ingenieurmäßiger Sicht analog zur Untersuchung mit lockeren Fertigungsrestriktionen am sinnvollsten. So würde die Optimierung näher an die günstigere Dreiecksform herankommen und über die gesamte Optimierung hinweg besser abschneiden. Eine Möglichkeit das zu erreichen könnte sein, die Anzahl der besten Entwürfe, die an die nächste Iteration übergeben werden, zu erhöhen. Dadurch bleibt der Entwurf basierend auf der UDSC-Heuristik in der ersten Iteration länger konkurrenzfähig.

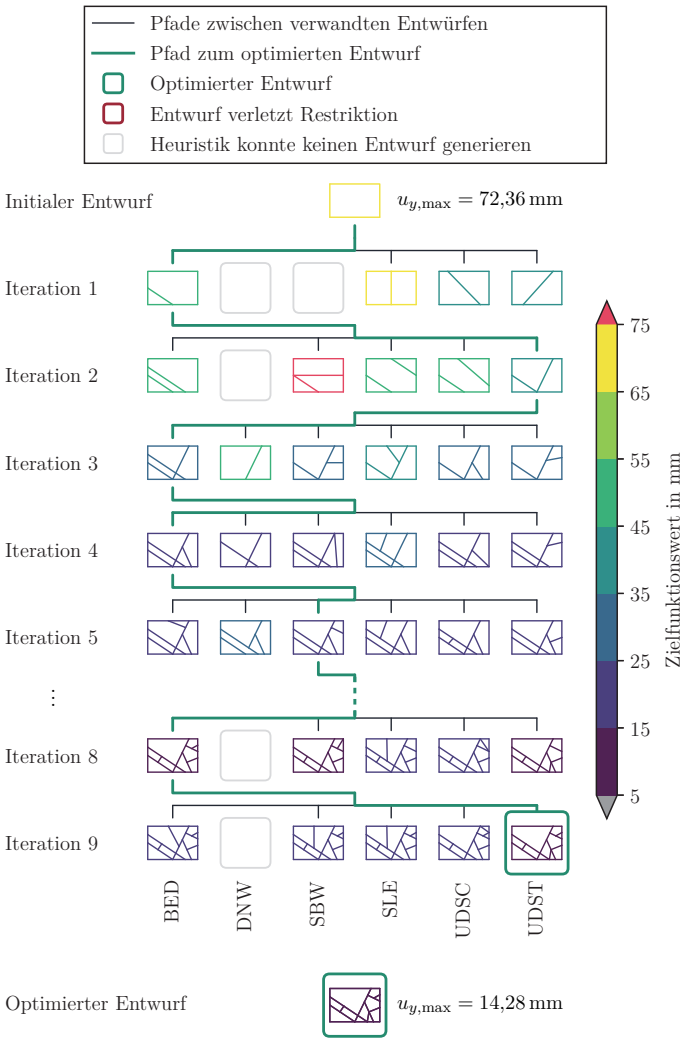
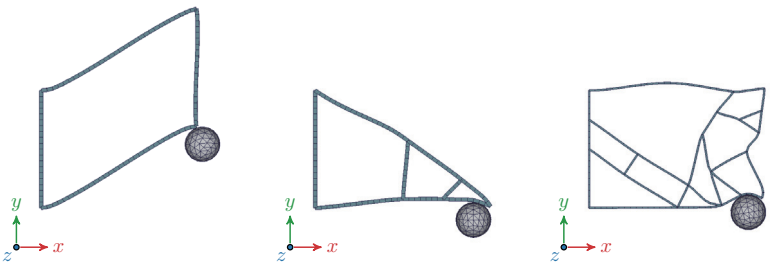


Abbildung 7–6: Entwurfsbaum der Optimierung der Impaktorverschiebung bei dem Rahmenmodell mit *inaktiver* RLS-Heuristik unter praxisnahen Fertigungsrestriktionen

Abbildung 7–7 zeigt abschließend das Deformationsverhalten des initialen Rahmenprofils und der optimierten Profile mit *aktiver* und *inaktiver* RLS-Heuristik. Der optimierte Rahmen aus der Optimierung mit aktiver RLS-Heuristik deformiert sich geringfügig. Die beiden Wände innerhalb des Dreieckrahmens stützen die äußeren Wände der Struktur sinnvoll ab. Konträr dazu ist das mechanische Verhalten des optimierten Rahmens ohne aktive RLS-Heuristik nicht erstrebenswert. Zwar schafft auch diese Struktur eine deutliche Verringerung der Impaktorverschiebung gegenüber dem Startentwurf, allerdings ist der optimierte Entwurf geprägt von starken lokalen Deformationen und ineffizient belasteten Wänden innerhalb des Rahmens. Durch die geringe Wandstärke von $t_{e_i}^{\{G_S\}} = 1{,}43\,\text{mm}$ schafft es der Impaktor weiter in die Struktur einzudringen, wodurch sich der zugehörige Zielfunktionswert verschlechtert.



(a) Startentwurf $(\cdot)$
$u_{y,\max} = 72{,}36\,\text{mm}$
$t = 13{,}65\,\text{ms}$
$t_{e_i}^{\{G_S\}} = 3\,\text{mm}$

(b) Optimierter Rahmen (RLS aktiv)
$u_{y,\max} = 7{,}82\,\text{mm}$
$t = 1{,}25\,\text{ms}$
$t_{e_i}^{\{G_S\}} = 2{,}98\,\text{mm}$

(c) Optimierter Rahmen (RLS inaktiv)
$u_{y,\max} = 14{,}28\,\text{mm}$
$t = 2{,}75\,\text{ms}$
$t_{e_i}^{\{G_S\}} = 1{,}43\,\text{mm}$

Abbildung 7–7: Deformationsbilder der initialen und optimierten Rahmenstrukturen unter praxisnahen Fertigungsrestriktionen mit repräsentativer Wanddicke zum Zeitpunkt der minimalen kinetischen Energie des Impaktors

Aktive Formoptimierung In dieser Untersuchung soll der Einfluss einer Formoptimierung zu Beginn und am Ende der Optimierung untersucht werden. Wie bereits angesprochen wurde, soll das die GHT-Optimierung mit inaktiver RLS-Heuristik zu Beginn in eine günstige Richtung lenken. Dadurch wird der Einsatz der RLS-Heuristik maßgeblich erschwert, da der zentrale Vorteil der diagonalen Kante zur Bildung des Dreieckprofils nicht mehr auftritt. Bei einer Optimierung mit aktiver Formoptimierung und lockeren Fertigungsrestriktionen aus den vorangegangenen Untersuchungen wird die RLS-Heuristik entlang des Pfades zur optimierten Struktur nicht aktiviert, weshalb diese Untersuchung hier nicht gezeigt wird.

Stattdessen wird die Optimierung mit aktiver Formoptimierung unter praxisnahen Fertigungsrestriktionen gezeigt. Die Optimierungsaufgabe lautet entsprechend analog zu der vorangegangenen Untersuchung:

$$
\begin{aligned}
\min \quad & u_{y,\max} := \max\, u_y(t), \\
\text{so dass} \quad & l_{e_i}^{\{G_S\}} \geq 15\,\text{mm}, \\
& d_{e_i,e_j}^{\{G_S\}} \geq 15\,\text{mm}, \\
& \alpha_{e_i,e_j}^{\{G_S\}} \geq 15^\circ, \\
& 1\,\text{mm} \leq t_{e_i}^{\{G_S\}} \leq 5\,\text{mm} \text{ und} \\
& m = m_{\text{init}} = 20{,}25\,\text{g}.
\end{aligned}
$$

Der zu dieser Optimierung zugehörige Ausschnitt des Entwurfsbaums wird in Abbildung 7–8 dargestellt. Mit insgesamt fünf Iterationen fällt diese Optimierung hinsichtlich der Anzahl der Iterationen signifikant kürzer aus als alle vorangegangenen Optimierungen. Durch die aktive Formoptimierung ist die Anzahl an Funktionsaufrufen mit 323 (304) dennoch hoch.

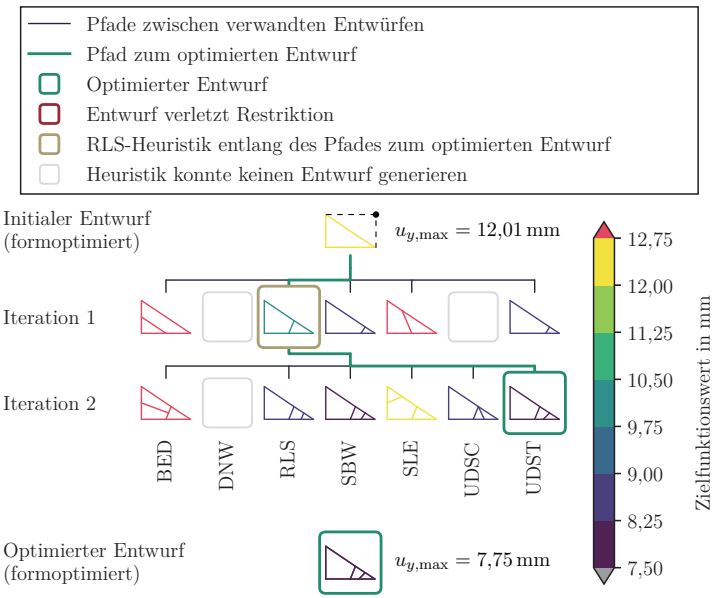


Abbildung 7–8: Entwurfsbaum der Optimierung der Impaktorverschiebung bei dem Rahmenmodell mit *aktiver* RLS-Heuristik und Formoptimierung

Die initiale Formoptimierung darf lediglich den oberen rechten Knoten innerhalb des zugrundeliegenden Rahmens verschieben. Dieser Knoten ist in Abbildung 7–8 im initialen Entwurf dargestellt. Dabei verbessert die Formoptimierung den Ausgangsentwurf von 72,36 mm auf 12,01 mm und benötigt dafür insgesamt 49 Funktionsaufrufe. Tabelle B–5 zeigt die Zusammenfassung der Optimierungsergebnisse. Durch die vorangeschaltete Formoptimierung entspricht der initiale Entwurf bereits einem Dreieckprofil.

Durch die Heuristiken wird in den Folgeiterationen die Steifigkeit der Struktur lokal am Auftreffpunkt des Impaktors erhöht. Dabei verbessert sich die y-Verschiebung des Impaktors auf 7,75 mm. Hierfür wird in der ersten Iteration die RLS-Heuristik eingesetzt. Es ist zu erkennen, dass die RLS-Heuristik eine Wand in die Struktur einbringt, die nicht mittig an die längste Außenwand des Dreieckprofils angebunden ist. Das liegt daran, dass dort der Vertex des initialen, nicht-formoptimierten Entwurfs durch die Formoptimierung positioniert wird. Die der RLS-Heuristik zugrundeliegende RL-Umgebung hat diesen Vertex entsprechend detektiert und für diese äußere Wand keine zusätzliche Kantenteilung benötigt. Viele andere Heuristiken erzeugen aber eine vergleichbare Form des Profilquerschnitts, weshalb der Mehrwert der RLS-Heuristik limitiert ist. In der finalen Formoptimierung wird trotz intensiver Suche keine verbesserte Form des Entwurfes gefunden. Für die Formoptimierung des finalen Entwurfs benötigt die GHT zusätzlich 181 Funktionsaufrufe. Die eingebrachten Wände sind also bereits durch die Topologievorschläge der Heuristiken naturgemäß günstig gelegen.

Der Entwurfsbaum der Referenzoptimierung mit inaktiver RLS-Heuristik wird in Abbildung 7–9 dargestellt. Das grundlegende Optimierungsverhalten in dieser Optimierung deckt sich mit der zugehörigen Optimierung mit aktiver RLS-Heuristik. Die GHT-Optimierung hat fünf Iterationen mit 310 Funktionsaufrufen, wobei die optimierte Struktur in der zweiten Iteration gefunden wird. Die genauen Zielfunktionswerte und Anzahl an Simulationsaufrufen in Abhängigkeit der Iterationen werden im Anhang in Tabelle B–6 aufgezeigt.

Diesmal schafft es die finale Formoptimierung eine geringfügig bessere Struktur zu finden, die die y-Verschiebung des Impaktors von 8,35 mm auf 8,27 mm verbessert. Damit ist der Zielfunktionswert trotz Formoptimierung schlechter, als der aus der analogen Optimierung mit aktiver RLS-Heuristik. Das stützt die Vermutung, dass die gefundene Struktur aus der vorangegangenen Untersuchung auch ohne finale Formoptimierung bereits sehr gut ist.

Das Deformationsverhalten des initialen, formoptimierten Rahmenprofils und der optimierten Profile mit *aktiver* und *inaktiver* RLS-Heuristik wird in Abbildung 7–10 aufgezeigt. Alle drei Struktur zeigen ein ähnliches Deformationsverhalten. Die beiden vollständig mit der GHT optimierten Entwürfe aus Abbildung 7–10b und Abbildung 7–10c werden durch zwei weitere Wände in der Nähe des Auftreffpunkts des Impaktors weiter versteift. Dadurch wird gleichzeitig die lange diagonale Außenwand des Dreieckprofils abgestützt.

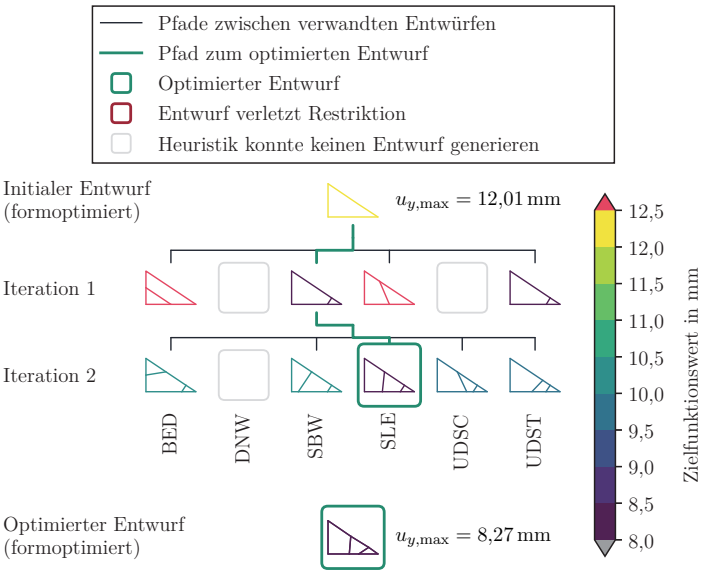
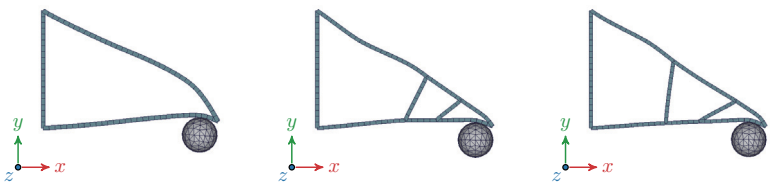


Abbildung 7–9: Entwurfsbaum der Optimierung der Impaktorverschiebung bei dem Rahmenmodell mit *inaktiver* RLS-Heuristik unter praxisnahen Fertigungsrestriktionen und Formoptimierung



(a) Startentwurf (formoptimiert)
$u_{y,\max} = 12{,}01\,\mathrm{mm}$
$t = 2{,}85\,\mathrm{ms}$
$t_{e_i}^{\{G_S\}} = 3{,}49\,\mathrm{mm}$

(b) Optimierter Rahmen (RLS aktiv)
$u_{y,\max} = 7{,}75\,\mathrm{mm}$
$t = 1{,}25\,\mathrm{ms}$
$t_{e_i}^{\{G_S\}} = 3{,}00\,\mathrm{mm}$

(c) Optimierter Rahmen (RLS inaktiv)
$u_{y,\max} = 8{,}27\,\mathrm{mm}$
$t = 1{,}35\,\mathrm{ms}$
$t_{e_i}^{\{G_S\}} = 2{,}89\,\mathrm{mm}$

Abbildung 7–10: Deformationsbilder der initialen, formoptimierten Rahmenstruktur und optimierten Rahmenstrukturen unter praxisnahen Fertigungsrestriktionen mit repräsentativer Wanddicke zum Zeitpunkt der minimalen kinetischen Energie des Impaktors

7.1.3 Minimierung der maximalen Impaktorbeschleunigung

Bei dieser Optimierungsaufgabe wird nur eine Optimierung mit lockeren Fertigungsrestriktionen gezeigt. Das liegt daran, dass mit den praxisnahen Fertigungsrestriktionen keine Optimierung gefunden wird, bei der die RLS-Heuristik zum Einsatz gekommen ist. Die konkrete Optimierungsaufgabe zur Minimierung der Impaktorbeschleunigung $\ddot{u}_{-y,\max}$ lautet:

$$
\begin{aligned}
\min \quad & \ddot{u}_{-y,\max} := \max -\ddot{u}_y(t), \\
\text{so dass} \quad & l_{e_i}^{\{G_S\}} \geq 5\,\text{mm}, \\
& d_{e_i,e_j}^{\{G_S\}} \geq 5\,\text{mm}, \\
& \alpha_{e_i,e_j}^{\{G_S\}} \geq 15^\circ, \\
& 1\,\text{mm} \leq t_{e_i}^{\{G_S\}} \leq 5\,\text{mm}, \\
& u_{y,\max} := \max\, u_y(t) \leq 50\,\text{mm und} \\
& \dot{u}_{y,\text{end}} := \dot{u}_y(t = t_{\text{end}}) \leq 0\,\text{mm/s}.
\end{aligned}
$$

Da die maximale Impaktorbeschleunigung $\ddot{u}_{-y,\max} := \max -\ddot{u}_y(t)$ in y-Richtung ermittelt wird, muss das Vorzeichen angepasst werden, da der Impaktor abgebremst wird. Die Beschleunigung wird mit dem Filter CFC-1000 aufbereitet, um hochfrequente Anteile im Signal zu entfernen. Als funktionale Restriktionen werden die Impaktorverschiebung $u_{y,\max}$ und die finale Impaktorgeschwindigkeit $\dot{u}_{y,\text{end}}$ restringiert. Die Struktur muss also eine Mindeststeifigkeit bemessen an der Impaktorverschiebung aufweisen. Gleichzeitig muss die Struktur die kinetische Energie des Impaktors vollständig aufnehmen, was durch einen Vorzeichenwechsel von einer positiven zu einer negativen Geschwindigkeit in y-Richtung am Ende der Simulation detektiert wird.

Für die folgenden Optimierungen der Impaktorbeschleunigung werden Parameter der SLE-Heuristik angepasst. Standardmäßig verbindet diese Heuristik die längsten Kanten in einem Graphen mittig voneinander. Hier werden die Parameter so eingestellt, dass die Heuristik eine Kante bei 40 % bzw. 60 % und die andere Kante bei 60 % bzw. 40 % ihrer Länge verbindet. Die kürzeste Verbindung zwischen den genannten Anbindungspositionen wird von der SLE-Heuristik realisiert. Der Grund für die Änderung der Anbindungsparameter ist, dass die RLS-Heuristik mit der standardmäßigen Einstellung nicht zum Einsatz kommt.

Für jeden der während der Optimierung vorgeschlagenen Entwürfe wird eine vollständige Dimensionierung durchgeführt. Auf eine Formoptimierung wird hier verzichtet, um den Fokus auf den direkten Einfluss der Heuristiken auf die Zielfunktion zu legen.

Abbildung 7–11 zeigt den Entwurfsbaum der Optimierung des Rahmenmodells bzgl. der Impaktorbeschleunigung unter lockeren Fertigungsrestriktionen. Insgesamt hat die GHT sechs Iterationen mit 1966 (1924) Funktionsaufrufen gebraucht, wobei die optimierte Struktur in der dritten Iteration gefunden wird. Durch die Dimensionierung jedes Entwurfs ist die Anzahl an Funktionsaufrufen entsprechend hoch im Vergleich zu den vorigen Optimierungen. Der Wert der Zielfunktion verbessert sich von 4661,42 m/s^2 auf 1302,34 m/s^2. Eine detaillierte numerische Übersicht über die Entwürfe entlang des Pfades zur optimierten Struktur wird im Anhang in Tabelle B–7 aufgezeigt.

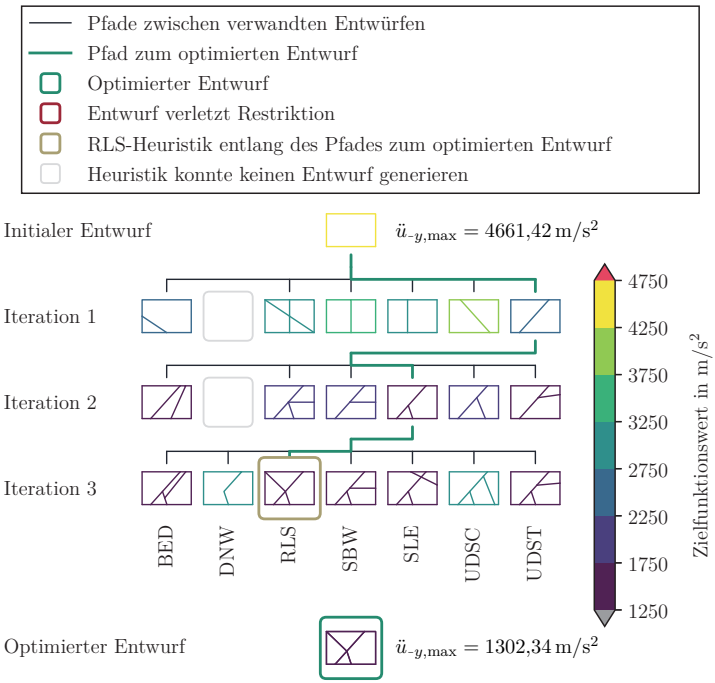


Abbildung 7–11: Entwurfsbaum der Optimierung der Impaktorbeschleunigung bei dem Rahmenmodell mit *aktiver* RLS-Heuristik unter lockeren Fertigungsrestriktionen

Keiner der gezeigten Entwürfe verletzt die Restriktionen. Alle Entwürfe entlang des Pfades zum optimierten Entwurf besitzen eine Verschiebung von $48\,\mathrm{mm} < u_{y,\max} < 50\,\mathrm{mm}$ und eine Endgeschwindigkeit $\dot{u}_{y,\mathrm{end}} \leq 0\,\mathrm{mm/s}$. Die Kombination aus den Heuristiken UDST, SLE und RLS versteift den linken Bereich der Struktur. Die kinetische Energie des Impaktors wird dann über die rechte Seite der Struktur abgebaut.

Auch für diese Optimierung wird eine Vergleichsoptimierung mit inaktiver RLS-Heuristik durchgeführt. Der Entwurfsbaum der Referenzoptimierung findet sich in Abbildung 7–12 wieder. Diese Optimierung hat insgesamt sieben Iterationen und 2045 Funktionsaufrufe benötigt, um mit der vierten Iteration die optimierte Struktur zu finden. Dabei verbessert sich die maximale Beschleunigung des Impaktors von 4661,42 m/s² aus dem initialen, dimensionierten Entwurf zu 1289,51 m/s². Damit ist dieser Zielfunktionswert mit ca. 1 % gegenüber der Optimierung mit RLS-Heuristik marginal besser. Analog zu den vorigen Untersuchungen werden die Optimierungsergebnisse im Anhang in Tabelle B–8 zusammengefasst.

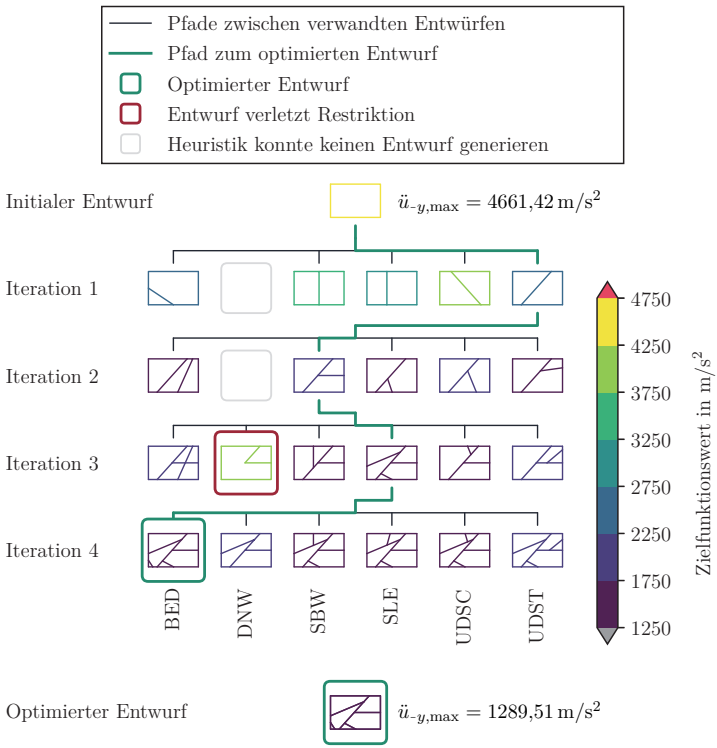


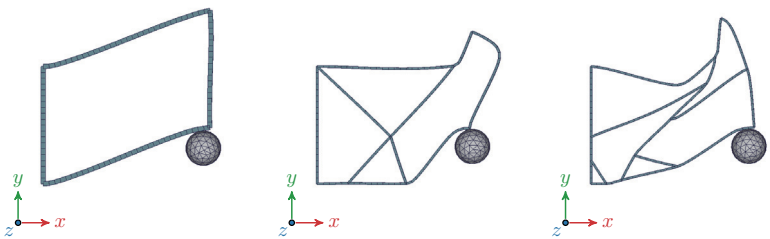
Abbildung 7–12: Entwurfsbaum der Optimierung der Impaktorbeschleunigung bei dem Rahmenmodell mit *inaktiver* RLS-Heuristik unter lockeren Fertigungsrestriktionen

In der ersten Iteration deckt sich die gewählte Heuristik UDST und der daraus abgeleitete Entwurf mit der vorangegangenen Optimierung mit aktiver RLS-Heuristik. Es fällt auf, dass der Entwurf der dritten Iteration auf Basis der SLE-Heuristik aus vier Wänden

besteht, wobei der übergeordnete Entwurf aus der zweiten Iteration aus lediglich zwei Wänden besteht. Das bedeutet, dass die SLE-Heuristik zwei Wände eingezogen hat, was aus einem fehlerhaften Verhalten der Heuristik resultiert. GHT-intern ist die Zuordnung von Kanten durch das Schneiden einer existierenden Kante beim Aufruf der Heuristik in der für diese Dissertation genutzten Implementierung der GHT durcheinander geraten. Lediglich die RLS-Heuristik darf korrekterweise bis zu zwei Wände in einem Heuristikaufruf in die Struktur einbringen.

Auch in dieser Optimierung besitzen alle Entwürfe entlang des Pfades zum optimierten Entwurf eine Verschiebung von $48\,\text{mm} < u_{y,\max} < 50\,\text{mm}$ und eine Endgeschwindigkeit $\dot{u}_{y,\text{end}} \leq 0\,\text{mm/s}$. Für diese Optimierung wäre es naheliegend, wenn die optimierte Struktur identisch mit der aus der Optimierung mit aktiver RLS-Heuristik wäre, da der hier gefundene Zielfunktionswert geringfügig besser ist und die RLS-Heuristik in der vorigen Optimierung nicht zum Einsatz hätte kommen müssen. Das ist aber nicht der Fall, da die grundsätzliche Aktivierung der RLS-Heuristik die entsprechenden Entwürfe in den vorangegangenen Iterationen verdrängt hat.

Abschließend werden in Abbildung 7–13 die Deformationen der initialen und optimierten Entwürfe aufgezeigt. Die Struktur aus der Optimierung mit aktiver RLS-Heuristik ist auf der linken Seite versteift und die rechte Seite der Struktur wird durch eine gezielte Beanspruchung der Wände deformiert, wodurch die niedrige Beschleunigung und die gewünschte Steifigkeit erreicht wird. Bei der optimierten Struktur mit inaktiver RLS-Heuristik wird das geforderte mechanische Verhalten aus einem komplexeren Zusammenspiel verschiedener Wände erreicht.



(a) Startentwurf (dimensioniert)
$\ddot{u}_{-y,\max} = 4661{,}42\,\text{m/s}^2$
$t = 9{,}40\,\text{ms}$
$t_{e_i}^{\{G_S\}} = 3{,}74\,\text{mm}$

(b) Optimierter Rahmen (RLS aktiv)
$\ddot{u}_{-y,\max} = 1302{,}34\,\text{m/s}^2$
$t = 10{,}40\,\text{ms}$
$t_{e_i}^{\{G_S\}} = 1{,}84\,\text{mm}$

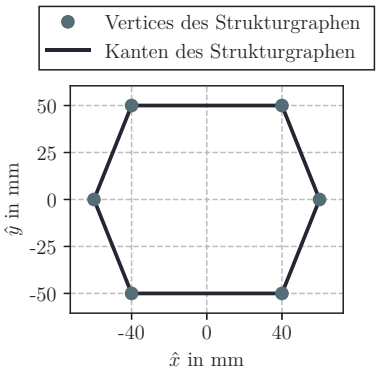
(c) Optimierter Rahmen (RLS inaktiv)
$\ddot{u}_{-y,\max} = 1289{,}51\,\text{m/s}^2$
$t = 9{,}65\,\text{ms}$
$t_{e_i}^{\{G_S\}} = 1{,}68\,\text{mm}$

Abbildung 7–13: Deformationsbilder der initialen, dimensionierten Rahmenstruktur und optimierten Rahmenstrukturen unter praxisnahen Fertigungsrestriktionen mit repräsentativer Wanddicke zum Zeitpunkt der minimalen kinetischen Energie des Impaktors

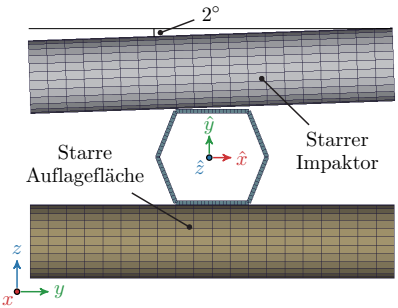
7.2 Biegeträger

7.2.1 Modellvorstellung

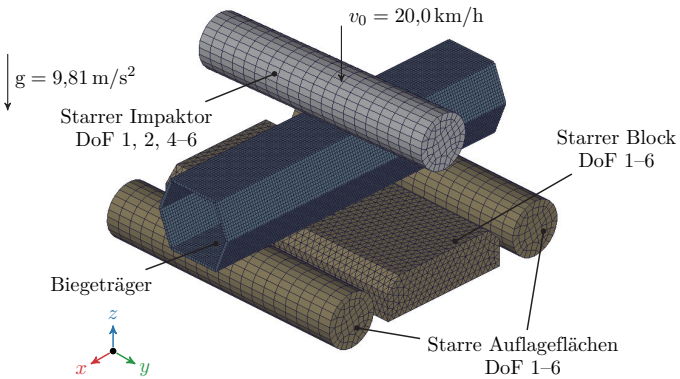
Als zweites Optimierungsmodell wird ein Biegeträger untersucht, welcher einen sechseckigen Profilquerschnitt besitzt. Das Optimierungsmodell ist angelehnt an das Modell aus der Veröffentlichung von Schneider (2023) und wird in Abbildung 7–14 vorgestellt.



(a) Strukturgraph des Biegeträgers in Profilkoordinaten



(b) Seitenansicht des Biegeträgers in der globalen yz-Ebene mit geneigtem Impaktor



(c) FE-Modell des Biegeträgers in isometrischer Ansicht inklusive Lastfall

Abbildung 7–14: Strukturgraph des Biegeträgers, welcher mittels GRAMB in das zugehörige FE-Modell übersetzt wird

Durch den sechseckigen Profilquerschnitt sollen auch Profile untersucht werden, welche in der ersten Iteration nicht von der RLS-Heuristik verarbeitet werden können. Das liegt daran, dass in der aktuellen Implementierung der RLS-Heuristik nur Agenten für 3- bis 5er-Zellen integriert sind. Zusätzlich leistet der sechseckige Querschnitt naturgemäß einen geringen Widerstand gegenüber dem Eindringen des Impaktors, so dass das Optimierungspotential groß ausfallen sollte.

Der Versuchsaufbau ist abgeleitet aus einem in Realität umgesetzten Fallturmversuch, bei dem ein Gewicht auf die Crashstruktur trifft. Das Profil besteht aus Aluminium (vgl. Abschnitt 5.4), hat eine Länge von 500 mm und hat bei einer initialen Blechdicke von 4,5 mm eine Masse von 2,281 kg. Der Biegeträger hat eine Elementkantenlänge von 4 mm und wird ohne Materialversagen berechnet. Der optimierte Entwurf soll durch Strangpressen des Aluminiumprofils herstellbar sein.

Als Impaktor dient ein starrer Zylinder mit einem Durchmesser von 80 mm und einem Gewicht von 250 kg, welcher um 2° um die globale x-Achse geneigt ist. Durch die Neigung des Impaktors soll das mögliche Spiel des geführten Impaktors im Versuchsstand abgebildet werden. Unter der in diesem Modell wirkenden Erdbeschleunigung $\mathrm{g} = 9{,}81\,\mathrm{m/s^2}$ besitzt der Impaktor eine initiale Geschwindigkeit von 20 km/h. Der starre Zylinder kann sich aufgrund von gesperrten Freiheitsgraden lediglich translatorisch entlang der globalen z-Richtung bewegen.

Zwei weitere starre Zylinder dienen als Auflagefläche für das Profil. Beide haben einen Durchmesser von 80 mm und sind fest eingespannt. Zwischen den beiden Auflageflächen befindet sich ein starrer Block, welcher ebenfalls fest eingespannt ist. Dieser ist in globale z-Richtung 10 mm tiefer positioniert als die Auflageflächen und erzwingt damit, dass das Profil nach einer gewissen Durchbiegung abgestützt wird.

7.2.2 Minimierung der maximalen Impaktorverschiebung

In dieser Optimierungsaufgabe soll die maximale Verschiebung des Impaktors in negative z-Richtung minimiert werden. Die konkrete Optimierungsaufgabe basierend auf praxisnahen Fertigungsrestriktionen lautet

$$
\begin{aligned}
\min \quad & u_{-z,\max} := \max\, -u_z(t), \\
\text{so dass} \quad & l_{e_i}^{\{G_\mathrm{S}\}} \geq 15\,\mathrm{mm}, \\
& d_{e_i,e_j}^{\{G_\mathrm{S}\}} \geq 15\,\mathrm{mm}, \\
& \alpha_{e_i,e_j}^{\{G_\mathrm{S}\}} \geq 15^\circ, \\
& 1\,\mathrm{mm} \leq t_{e_i}^{\{G_\mathrm{S}\}} \leq 5\,\mathrm{mm} \text{ und} \\
& m = m_\mathrm{init} = 2{,}281\,\mathrm{kg}.
\end{aligned}
$$

Wie in allen bisher durchgeführten Optimierungen wird zunächst die Optimierung mit aktiver RLS-Heuristik durchgeführt und anschließend mit einer Optimierung mit inaktiver RLS-Heuristik verglichen. Die Auswerteebene für die RLS-Heuristik zum Aufbau des Evaluationsgraphen wird mittig vom Sechseckprofil bei $\hat{z} = 250\,\text{mm}$ platziert. Das ist dort, wo der Impaktor auf die Struktur trifft.

In Abbildung 7–15 wird ein Ausschnitt des Entwurfsbaums für die Optimierung mit aktiver RLS-Heuristik gezeigt. Die GHT hat insgesamt 13 Iterationen und 293 (274) Funktionsaufrufe benötigt, bis das Abbruchkriterium gegriffen hat. Der optimierte Entwurf stammt aus der zehnten Iteration. Die Optimierung hat es geschafft, die Verschiebung des Impaktors in negative z-Richtung von 60,70 mm auf 30,02 mm zu reduzieren.

In der ersten Iteration wird die Heuristik SLE aktiviert, welche die beiden längsten Wände im Querschnitt mit einer neuen Wand verbindet. Die Verbindung wird nicht mittig, sondern in der linken Hälfte der Struktur eingebracht. Dort trifft der geneigte Impaktor zuerst auf die Struktur.

Die Heuristiken UDST und UDSC stützen in der zweiten und dritten Iteration die von der SLE-Heuristik eingezogene Wand ab und versteifen zusätzlich den initialen Auftreffpunkt des geneigten Impaktors. Es fällt auf, dass die eingezogene Wand aus der zweiten Iteration nicht mehr in dem optimierten Entwurf vorhanden ist. Diese wird in der achten Iteration, welche hier nicht visualisiert wird, durch die Heuristik DNW herausgelöscht.

In der vierten Iteration wird die RLS-Heuristik aktiviert. Diese bringt eine einzelne weitere Wand analog zu der Wand aus der ersten Iteration auf der rechten Seite des Profils ein. Dadurch kann diese Wand den Impaktor auch rechtsseitig abstützen. Auch diese Wand wird durch die UDST-Heuristik weiter abgestützt, wobei die abstützende Wand in der zehnten Iteration durch die DNW-Heuristik wieder entfernt wird.

Der optimierte Entwurf beinhaltet zusätzlich eine horizontale Wand im unteren Bereich der Struktur, welche die vertikalen Wände abstützt. Diese Wand stammt von der SLE-Heuristik aus der sechsten Iteration, welche aus Platzgründen nicht im Entwurfsbaum dargestellt ist. Eine weitere nahezu horizontale Wand wird innerhalb der beiden vertikalen Wände im oberen Bereich des Profils eingezogen, um die beiden tragenden vertikalen Wände weiter abzustützen. Diese Wand wird von der Heuristik UDSC in der neunten Iteration in das Profil eingebracht.

In Tabelle B–9 wird eine Zusammenfassung der Optimierungsergebnisse gegeben. Daraus geht hervor, dass ab der sechsten Iteration keine Funktionsaufrufe mehr der RLS-Heuristik entstammen. Zwar findet die Heuristik prinzipiell Zellen, doch sind die Strukturen durch den Prozess der Kantenteilung als Resultat der praxisnahen Fertigungsrestriktionen nicht mehr herstellbar.

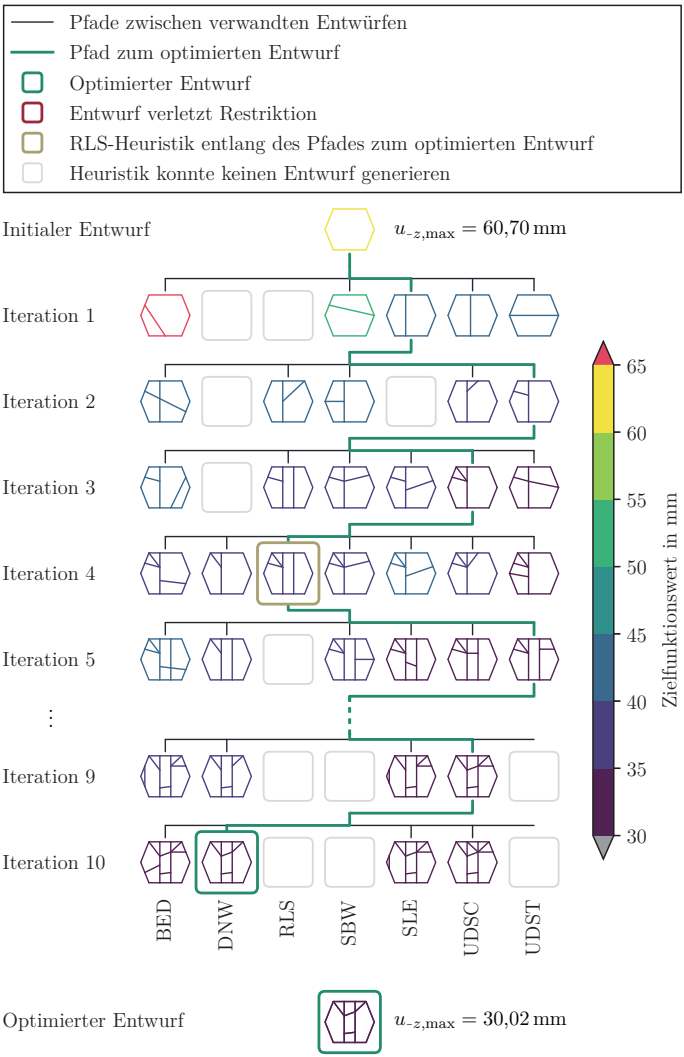
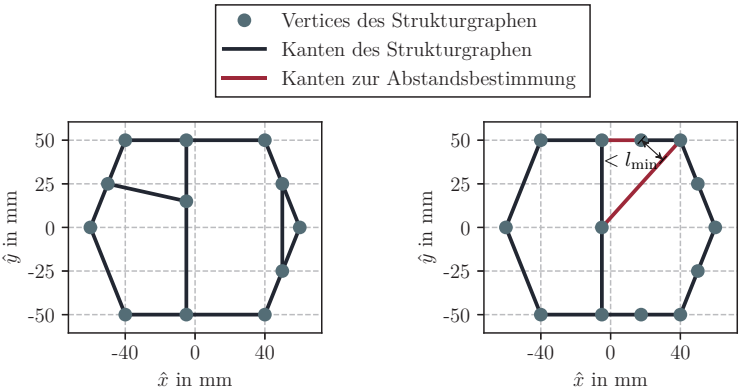


Abbildung 7–15: Entwurfsbaum der Optimierung der Impaktorverschiebung des Biegeträgers mit *aktiver* RLS-Heuristik

Diese Optimierung bietet sich folglich an, zwei exemplarische Entwürfe aus der Optimierung bezüglich der Einhaltung der Fertigungsrestriktionen zu analysieren. Diese sind in Abbildung 7–16 dargestellt. Einer der Entwürfe ist herstellbar, der andere ist nicht herstellbar.



(a) Strukturgraph der fertigungsgerechten Struktur

(b) Strukturgraph der nicht herstellbaren Struktur aus Topologiemodifikation mit RLS-Heuristik

Abbildung 7–16: Fertigungsgerechte und nicht herstellbare Struktur im Sinne der definierten Fertigungsrestriktionen

Es fällt direkt auf, dass der Strukturgraph der fertigungsgerechten Struktur (Abbildung 7–16a) aufgrund der kleinen Kammer am rechten Rand deutlich schwieriger zu fertigen ist als die rechte Struktur (Abbildung 7–16b) basierend auf dem dargestellten Strukturgraphen. Dieses unintuitive Verhalten wird hier als Anlass genutzt, um eine zentrale Schwierigkeit der RLS-Heuristik herauszuarbeiten. Durch den Kantenteilungsprozess wird die Herstellbarkeit der Struktur erschwert. Zum einen werden die Kantenlängen halbiert, ohne die Form und Topologie des Graphen und der abgeleiteten Struktur zu ändern. Häufig wird dabei der kritische Grenzwert der minimalen erlaubten Kantenlänge unterschritten. Zum anderen kann es passieren, dass wie in Abbildung 7–16b dargestellt wird, die minimal geforderten Wand- bzw. Kantenabstände unterschritten werden. Das liegt daran, dass bei der Überprüfung des Abstands von zwei Kanten bereits verbundene Kanten ausgeschlossen werden. Durch den Kantenteilungsprozess entsteht eine neue, unverbundene Kante. Diese Kante wird dann näher an der zu vergleichenden Kante detektiert, obwohl die Form und Topologie der aus dem Strukturgraphen abgeleiteten Struktur unverändert ist.

Der Entwurfsbaum der Referenzoptimierung mit inaktiver RLS-Heuristik wird in Abbildung 7–17 dargestellt. Mit insgesamt sieben Iterationen und 127 Funktionsaufrufen fällt diese Optimierung kürzer aus als die Optimierung mit aktiver RLS-Heuristik. Die optimier-

te Struktur wird in der fünften Iteration gefunden. Der Zielfunktionswert verbessert sich von 60,70 mm auf 34,02 mm. Tabelle B–10 stellt eine iterationsweise Zusammenfassung der Optimierung dar.

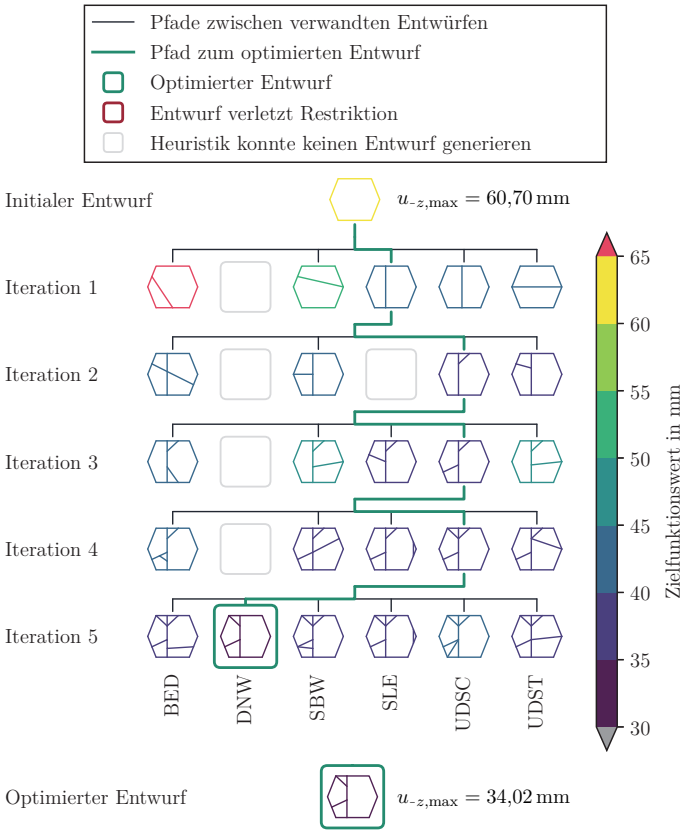
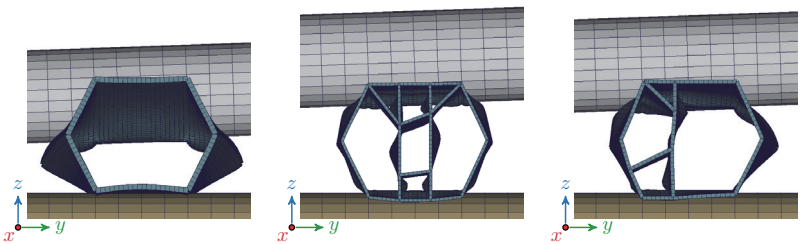


Abbildung 7–17: Entwurfsbaum der Optimierung der Impaktorverschiebung des Biegeträgers mit *inaktiver* RLS-Heuristik

Die Wahl der SLE-Heuristik in der ersten Iteration deckt sich mit der vorigen Optimierung. Im weiteren Verlauf der vorliegenden Optimierung zeigt sich, dass die Entwürfe entlang des Pfades zur optimierten Struktur lediglich den linken Bereich des Sechseckprofils ausnutzen. Dort findet auch der initiale Kontakt zum Impaktor statt. Die vertikal eingezogene Wand wird gegen Beulen von zwei weiteren Wänden abgestützt. Auch in dieser Optimierung wird die in der zweiten Iteration mit der Heuristik UDSC eingezogene Wand wieder herausgelöscht.

In Abbildung 7–18 werden die Deformationsbilder des Startentwurfs und der optimierten Entwürfe mit aktiver und inaktiver RLS-Heuristik aufgezeigt. Wie zu erwarten war, zeigt sich auch in dem Deformationsverhalten des Startentwurfs, dass das leere sechseckige Profil nur eine geringe Steifigkeit unter lateraler Belastung aufweist. Durch die versteifenden vertikalen Wände und deren Abstützungen schaffen es beide optimierten Strukturen ihre Form gut zu erhalten.



(a) Startentwurf $(\cdot)$
$u_{-z,\max} = 60{,}70\,\text{mm}$
$t = 25{,}53\,\text{ms}$
$t_{\dot{e}_i}^{\{G_S\}} = 4{,}50\,\text{mm}$

(b) Optimierter Biegeträger (RLS aktiv)
$u_{-z,\max} = 30{,}02\,\text{mm}$
$t = 9{,}40\,\text{ms}$
$t_{\dot{e}_i}^{\{G_S\}} = 2{,}38\,\text{mm}$

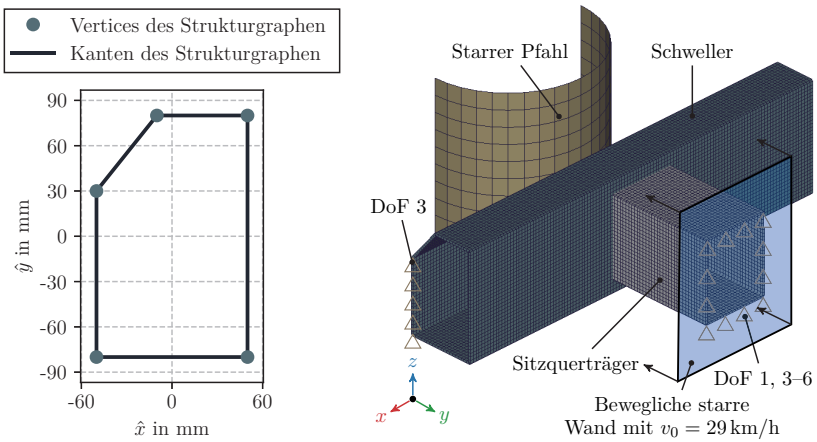
(c) Optimierter Biegeträger (RLS inaktiv)
$u_{-z,\max} = 34{,}02\,\text{mm}$
$t = 11{,}55\,\text{ms}$
$t_{\dot{e}_i}^{\{G_S\}} = 3{,}06\,\text{mm}$

Abbildung 7–18: Deformationsbilder der initialen und optimierten Biegeträger mit repräsentativer Wanddicke zum Zeitpunkt der minimalen kinetischen Energie des Impaktors

7.3 Schwellerausschnitt

7.3.1 Modellvorstellung

Als letztes Modell wird ein Ausschnitt eines Schwellers betrachtet, welcher in Abbildung 7–19 dargestellt wird. Eingeführt wurde das Modell von Ortmann (2015). Der zu optimierende Schwellerausschnitt mit einer Elementkantenlänge von 5 mm besteht aus Aluminium (vgl. Abschnitt 5.4) und hat eine initiale Wandstärke von 3,5 mm. Materialversagen wird in diesem Modell nicht berücksichtigt. Das Profil ist 600 mm lang und hat eine Masse von 2,801 kg. Die Energie einer sich bewegenden starren Wand wird über den Sitzquerträger in den Schweller eingebracht. Diese bewegt sich in globale y-Richtung, hat eine initiale Geschwindigkeit von $v_0 = 29\,\text{km/h}$ und eine Masse von 85 kg. Der Schwellerausschnitt trifft auf einen sich nicht bewegenden, starren Pfahl mit einem Durchmesser von 255 mm. An den markierten Dreiecken sind angrenzende FE-Knoten eingespannt. Dazu gehören Teile beider Profilenden, die in ihrer globalen z-Verschiebung verhindert sind. Das freie Ende des Sitzquerträgers darf sich lediglich translatorisch in z-Richtung bewegen.



(a) Strukturgraph des Schwellerausschnitts in Profilkoordinaten

(b) FE-Modell des Schwellerausschnitts in isometrischer Ansicht

Abbildung 7–19: Strukturgraph des Schwellerausschnitts, welcher mittels GRAMB in das zugehörige FE-Modell übersetzt wird

7.3.2 Minimierung der Pfahlintrusion

Entsprechend der Funktionsweise der RLS-Heuristik soll in dieser Untersuchung die Steifigkeit des Schwellers maximiert werden. Die Auswerteebene für den Aufbau des Evaluationsgraphen wird mittig von dem Profil bei $\hat{z} = 300\,\mathrm{mm}$ gewählt. Als Zielfunktion wird dafür die maximale Verschiebung der starren Wand in negative y-Richtung $u_{-y,\max}$ betrachtet. Analog zu vorangegangenen Optimierungen soll auch hier die Strukturmasse konstant bleiben. Die Fertigungsrestriktionen decken sich mit denen aus den Untersuchungen mit praxisnahen Fertigungsrestriktionen, um den Anwendungsbezug der Optimierungsaufgabe zu gewährleisten. Mathematisch lässt sich die Optimierungsaufgabe formulieren als

$$
\begin{aligned}
\min \quad & u_{-y,\max} := \max\, -u_y(t), \\
\text{so dass} \quad & l_{e_i}^{\{G_\mathrm{S}\}} \geq 15\,\mathrm{mm}, \\
& d_{e_i,e_j}^{\{G_\mathrm{S}\}} \geq 15\,\mathrm{mm}, \\
& \alpha_{e_i,e_j}^{\{G_\mathrm{S}\}} \geq 15^\circ, \\
& 1\,\mathrm{mm} \leq t_{e_i}^{\{G_\mathrm{S}\}} \leq 5\,\mathrm{mm} \text{ und} \\
& m = m_\mathrm{init} = 2{,}801\,\mathrm{kg}.
\end{aligned}
$$

Analog zu den bisher durchgeführten Optimierungen wird die Optimierung mit aktiver RLS-Heuristik mit einer Optimierung mit inaktiver RLS-Heuristik verglichen. Zum Aufbau des Evaluationsgraphen wird erneut die Mitte des Profils entlang der Extrusionsrichtung bei $\hat{z} = 300\,\text{mm}$ gewählt, da dort der starre Pfahl auf das Schwellermodell trifft. Der Entwurfsbaum für die Optimierung mit aktiver RLS-Heuristik wird in Abbildung 7–20 dargestellt. Insgesamt hat die Optimierung 225 (189) Funktionsaufrufe benötigt, wovon 61 (25) auf die RLS-Heuristik zurückzuführen sind. In Tabelle B–11 wird eine Zusammenfassung der Optimierungsergebnisse gegeben.

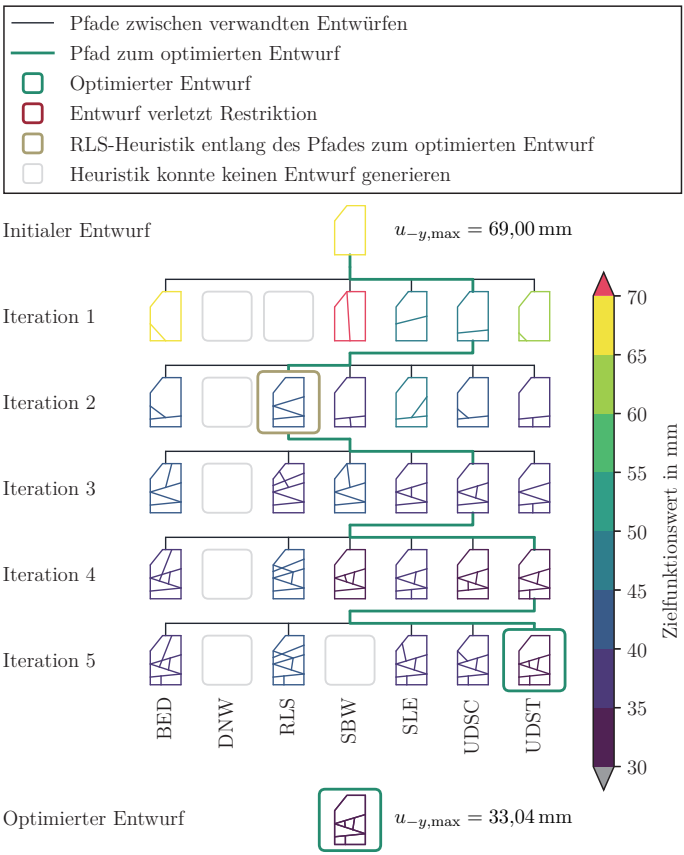


Abbildung 7–20: Entwurfsbaum der Optimierung der Pfahlintrusion des Schwellerausschnitts mit *aktiver* RLS-Heuristik

In der ersten von acht Iterationen wird entlang des Pfades zur optimierten Struktur die Heuristik UDSC aktiviert. Diese Heuristik zieht im unteren Bereich der Struktur eine näherungsweise horizontale Wand ein, um der Bewegungsrichtung der Sitzquerträgers und der starren Wand entgegenzuwirken und somit die Steifigkeit der Struktur zu vergrößern.

Ebenso hat die RLS-Heuristik in der ersten Iteration scheinbar keinen Entwurf gefunden. Das ist darauf zurückzuführen, dass dieser Entwurf identisch mit dem der SLE-Heuristik ist. Bei identischen Entwürfen sortiert die GHT doppelte Entwürfe aus, um unnötige Funktionsaufrufe einzusparen.

In der zweiten Iteration wird die Heuristik RLS aktiviert und zieht in einer Iteration zwei Wände in die Struktur ein. Dabei bilden die Wände einen Dreiecksquerschnitt. Die Verschiebung der starren Wand ist hier bereits von 69,00 mm auf 40,69 mm reduziert. Damit ist der Entwurf allerdings nur der drittbeste Entwurf in der zweiten Iteration. Der beste Entwurf in der zweiten Iteration entstammt den Heuristiken UDSC und UDST mit einem Zielfunktionswert von 39,56 mm.

Die in den folgenden drei Iterationen eingezogenen Wände stützen die bereits eingezogenen Wände weiter ab, sodass der optimierte Zielfunktionswert 33,04 mm entspricht. In diesen Iterationen versucht die RLS-Heuristik je zwei sich schneidende Wände in der oberen Hälfte des Schwellers einzubringen. Eine Ursache dafür liegt an der großen Querschnittsfläche, die von der zugehörigen Zelle aufgespannt wird. Dadurch wird der Einfluss der Zelle groß geschätzt und im Zellauswahlprozess bevorzugt. Dennoch schneiden andere Heuristiken der GHT besser ab und die Vorschläge der RLS-Heuristik werden langfristig verdrängt.

Als Referenzoptimierung mit inaktiver RLS-Heuristik wird in Abbildung 7–21 der zugehörige Entwurfsbaum vorgestellt. In dieser Optimierung hat die GHT insgesamt zwölf Iterationen berechnet und dabei den optimierten Entwurf in der neunten Iteration gefunden. Die Zielfunktion wird in dieser Optimierung von 69,00 mm auf 31,22 mm verbessert und ist damit 1,82 mm besser als die Optimierung mit aktiver RLS-Heuristik. In Tabelle B–12 werden die Optimierungsergebnisse zusammengefasst und übersichtlich dargestellt.

Eine Ursache für den besseren Zielfunktionswert gegenüber der Optimierung mit aktiver RLS-Heuristik ist, dass der Entwurf der RLS-Heuristik in der zweiten Iteration mittelfristig für die GHT sinnvoll erscheint. So beinhalten die drei besten Entwürfe in der dritten Iteration die RLS-Heuristik. Langfristig haben allerdings Entwürfe aus der Optimierung mit inaktiver RLS-Heuristik besser abgeschnitten.

Der Optimierungsverlauf ist aufgrund der fehlenden dreieckigen Kammer durch die RLS-Heuristik grundsätzlich anders. Trotzdem finden sich grobe Gemeinsamkeiten darin, dass in beiden Optimierungen horizontale Wände eingezogen werden, um der Verschiebung der starren Wand entgegenzuwirken.

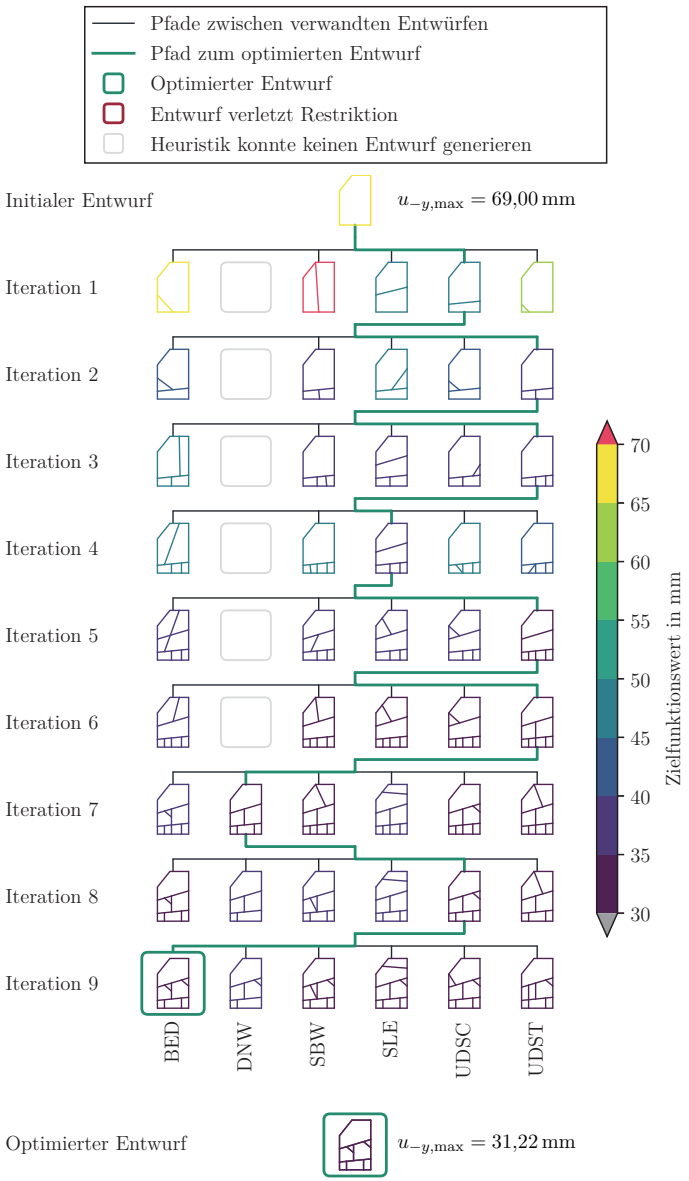
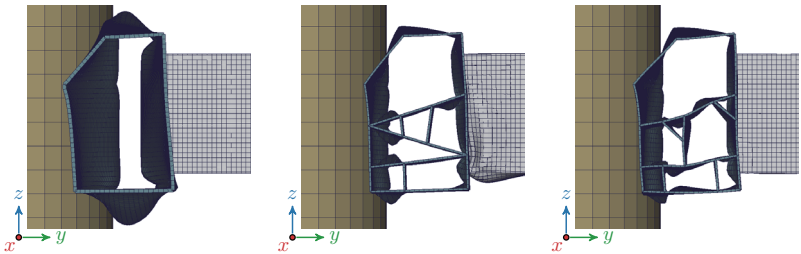


Abbildung 7–21: Entwurfsbaum der Optimierung der Pfahlintrusion des Schwellerausschnitts mit *inaktiver* RLS-Heuristik

Zum Abschluss dieser Untersuchung wird in Abbildung 7–22 das Deformationsverhalten der Schwellerausschnitte für den Startentwurf und die beiden optimierten Varianten aufgezeigt.



(a) Startentwurf
$(\cdot)$
$u_{-y,\max} = 69{,}00\,\text{mm}$
$t = 16\,\text{ms}$
$t_{e_i}^{\{G_S\}} = 3{,}50\,\text{mm}$

(b) Optimierter Schweller
(RLS aktiv)
$u_{-y,\max} = 33{,}04\,\text{mm}$
$t = 8\,\text{ms}$
$t_{e_i}^{\{G_S\}} = 1{,}94\,\text{mm}$

(c) Optimierter Schweller
(RLS inaktiv)
$u_{-y,\max} = 31{,}22\,\text{mm}$
$t = 7\,\text{ms}$
$t_{e_i}^{\{G_S\}} = 1{,}99\,\text{mm}$

Abbildung 7–22: Deformationsbilder der initialen und optimierten Schwellerausschnitte mit repräsentativer Wanddicke zum Zeitpunkt der minimalen kinetischen Energie des Sitzquerträgers

Durch die fehlende Abstützung innerhalb des Schwellerausschnitts ist die Steifigkeit des Startentwurfs gering, was sich entsprechend in dem hohen Zielfunktionswert widerspiegelt. Bei dem optimierten Schwellerausschnitt aus der Optimierung mit aktiver RLS-Heuristik fängt der Sitzquerträger an zu beulen. Das bedeutet, dass der Schweller so steif ist, dass die dahinterliegende Komponente einen Teil der kinetischen Energie der starren Wand absorbiert. Dieser Effekt wird aufgrund der Formulierung der Zielfunktion nicht detektiert, da die Verschiebung an der starren Wand ermittelt wird. Dadurch trägt der Sitzquerträger zu dem schlechteren Zielfunktionswert bei. Legt man beide optimierten Schwellerdeformationen linksbündig am Pfahl übereinander erkennt man, dass der Schweller aus der Optimierung mit aktiver RLS-Heuristik trotz höherem $u_{-y,\max}$ weniger stark deformiert wird und somit steifer ist als der Entwurf aus der Optimierung mit inaktiver RLS-Heuristik. Dieses Phänomen wird in Abbildung 7–23 visualisiert.

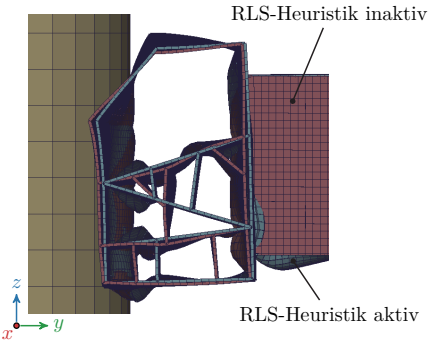


Abbildung 7–23: Überlagerung der optimierten Entwürfe aus den Optimierungen mit aktiver und inaktiver RLS-Heuristik zur Visualisierung der Steifigkeit der Schwellerausschnitte

7.4 Diskussion der Heuristikperformance

In diesem Abschnitt werden die Optimierungsuntersuchungen damit abgeschlossen, dass eine qualitative Bewertung der Performance der RLS-Heuristik erfolgt und die Stärken und Schwächen dieser in praktischen GHT-Optimierungen herausgearbeitet werden.

Für alle Optimierungen muss berücksichtigt werden, dass die RLS-Heuristik bis zu zwei Wände in die Strukturen einziehen darf. Andere Heuristiken bringen stets genau eine Wand in die Struktur ein. Unter diesem Gesichtspunkt hat die RLS-Heuristik einen Vorteil gegenüber den existierenden Heuristiken, was bei der Bewertung berücksichtigt werden muss. Zusätzlich erzwingt das sequentielle Einbringen von Wänden durch die RLS-Heuristik, dass die Optimierungsdauer ansteigt. Andere Heuristiken, welche nur eine FE-Simulation benötigen, warten auf die RLS-Heuristik, die nach jeder eingezogenen Wand eine neue FE-Simulation durchführen muss. Beides muss bei der Analyse der Optimierungen beachtet werden.

Es fällt auf, dass die RLS-Heuristik in allen Optimierungen stets nur einmal im Pfad zum optimierten Entwurf aktiviert wird. Meistens wird die RLS-Heuristik in einer frühen Iteration eingesetzt. Das ergibt deshalb Sinn, da in den frühen Iteration die gefundenen Zellen größer sind und der Gesamteinfluss auf die Struktur damit auch größer ist. Gleichzeitig ist die Wahrscheinlichkeit groß, dass bei kleineren Zellen in späteren Iterationen die Fertigungsrestriktionen durch den Kantenteilungsprozess nicht mehr eingehalten werden können. Durch den tendenziell frühen Einsatz der RLS-Heuristik in den Optimierungen bestimmt diese maßgeblich den grundlegenden Aufbau und das mechanische Verhalten des optimierten Entwurfs. Bei den durchgeführten Optimierungen, die ausschließlich auf Heuristikaktivierung basieren und keine Formoptimierung oder Dimensionierung beinhalten, sind die optimierten Entwürfe auf Basis RLS-Heuristik mechanisch effizienter.

Die Strukturen sind einfacher und sie haben meist einen besseren Zielfunktionswert. Entsprechend ist der Nutzen der RLS-Heuristik für Optimierungsaufgaben, in denen die Steifigkeit der Struktur maximiert werden soll, auch über das Rahmenmodell hinaus nachweisbar.

8 Zusammenfassung

In dieser Dissertation wurde eine auf RL basierende, lokal agierende Heuristik für die GHT entwickelt, um neue und diverse Strukturentwürfe in Crashoptimierungen zu generieren und somit die Optimierung auf Kosten von zusätzlichen Funktionsaufrufen zu potentiell besseren Optima zu führen. Dabei wurden die in Abschnitt 1.2 definierten Forschungsfragen aufgegriffen und diskutiert. Im Folgenden werden die gefundenen Antworten zusammengefasst.

Welche zentralen Schnittstellen hat die GHT, bei denen KI-Methoden eingesetzt werden können?

Für die GHT konnten drei zentrale Schnittstellen extrahiert werden.

- Die erste Schnittstelle ist der Optimierungsablauf der GHT an sich. Hier kann in diesen eingegriffen werden, sodass einzelne Iterationen oder im Idealfall gleich die gesamte Optimierung übersprungen wird. In diesen ausgelassenen Iterationen kann dann die KI eingreifen und einen Vorschlag für einen Entwurf direkt aus den bereits vorliegenden Optimierungsdaten generieren. So kann die Optimierungszeit einer Optimierungsaufgabe durch das Einsparen zahlreicher Funktionsaufrufe reduziert werden.
- Als zweite Schnittstelle werden die Crashberechnungen innerhalb der GHT betrachtet. Mit Methoden der KI können Deformationsfelder nach dem Anrechnen einer Simulation vorhergesagt werden. Für die sinnvolle Verwendung in der GHT müssen dann aber auch andere (Feld-)größen vorhergesagt werden können. Dadurch kann Rechenzeit eingespart werden, da frühzeitig abgeschätzt werden kann, ob ein Entwurf die gewünschte Performance erbringt.
- Die dritte Schnittstelle interagiert mit den oder erweitert die Heuristiken der GHT. Zum einen ist es möglich, dass eine KI einen vorliegenden Entwurf analysiert und darauf basierend vorhersagt, welche der Heuristiken die geeignetste für die nächste Iteration ist. So muss nur der Entwurf dieser Heuristik berechnet werden. Auch mit diesem Ansatz können weitere Funktionsaufrufe eingespart werden, da alle anderen Entwürfe aus Heuristikvorschlägen nicht berechnet werden müssen. Ein anderer Ansatz, welcher mit dieser Dissertation verfolgt wurde, ist die Implementierung einer neuen KI-basierten Heuristik. Dabei ist die Aufgabe der Heuristik neue Entwurfsvorschläge konkurrierend zu den bestehenden Heuristiken zu generieren und gegen diese anzutreten.

Welche dieser Schnittstellen können einen praktischen Mehrwert für die Entwicklung der GHT generieren?

Zwar sind die genannten Ansätze zur Einsparung von Rechenzeit einer GHT-Optimierung für den praktischen Gebrauch grundsätzlich erstrebenswert, jedoch helfen sie nicht dabei die Methode an sich zu verbessern. Für diese Dissertation wurde deshalb der Fokus gewählt, mit einer neuen KI-basierten Heuristik neue Entwürfe zu generieren, um die existierenden Heuristiken zu ergänzen und die Methode der GHT zu erweitern und zu verbessern.

Wie lässt sich eine konkrete RL-Methodik zur Unterstützung der GHT formulieren?

Das zentrale Konzept der entwickelten Heuristik besteht darin, dass ein RL-Agent das mechanische Verhalten einer Zelle eines Strukturgraphen analysiert und darauf basierend eine Entscheidung trifft, welche Vertices der Zelle miteinander verbunden werden sollten, um die Steifigkeit der Zelle zu erhöhen. Dazu wurde der (modifizierte) Zellen- und der Evaluationsgraph eingeführt. Der modifizierte Zellengraph ermöglicht eine konsistente Beschreibung der Zelle in Vektorform und der Evaluationsgraph stellt die aus der FE-Simulation abgeleiteten Strukturantworten in einem Querschnitt des simulierten Extrusionsprofils bereit. Aus den beiden Graphen kann eine einheitlich aufgebaute Observation extrahiert werden, welche für Zellen mit fixer Seitenanzahl unabhängig von der Geometrie und Topologie innerhalb der Zelle ist. Die Observation beinhaltet verschiedene geometrische und mechanische Größen, welche beispielsweise die Deformation der Zelle beschreiben. Verschiedene Agenten für unterschiedliche Zelltypen wurden mittels PPO-Algorithmus auf vorberechneten, randomisierten Datensätzen trainiert, um den Ressourcenaufwand für ein erstes Training gering zu halten. Dabei konnten die Agenten bis zu zwei Kanten erfolgreich in die Zelle einbringen. Ein weiteres Training auf neuen, ungesehenen und randomisierten Daten mit bis zu maximal drei einziehbaren Kanten konnte die Performance der Agenten nicht weiter verbessern, sodass die Agenten der vorigen Untersuchung als finale Agenten herangezogen wurden. Diese Agenten mitsamt der zugrundeliegenden Umgebung wurden als RLS-Heuristik in die GHT integriert.

Inwiefern eignet sich RL gegenüber anderen KI-Methoden im Kontext der GHT?

RL wurde in dieser Dissertation als primäre ML-Methode eingesetzt. Der Vorteil gegenüber Methoden des SLs ist, dass zum Trainieren des zugrundeliegenden ML-Modells die Optima nicht bekannt sein müssen. Es wird also kein Datensatz benötigt, der ausschließlich aus den besten oder zumindest guten Entwürfen im Sinne einer Zielfunktion besteht. Die Bestimmung solcher Entwürfe würde nicht nur hochgradig ressourcenintensiv sein, sondern auch höchst ineffizient, da die weniger guten Entwürfe nicht zum Training des ML-Modells als Negativbeispiel herangezogen werden. Zwar ist auch RL bekannt für eine

hohe Sample-Komplexität, allerdings kann das ML-Modell bzw. der Agent durch das Trial-And-Error-Prinzip sich selbstständig und zielgerichtet ohne vorherige Informationen an die Generierung optimaler Entwürfe herantasten. Ebenso kann RL gegenüber dem SL und dem UL auch basierend auf einer Folge von Entscheidungen lernen.

Wie gut kann die in dieser Arbeit umgesetzte RL-Methodik die GHT praktisch unterstützen?

Diese Dissertation konnte in unterschiedlichen praktischen Optimierungsbeispielen mit der GHT zeigen, dass der Optimierungsablauf von der entwickelten RLS-Heuristik profitieren kann. Insbesondere die Optimierungsaufgaben bei denen die Steifigkeit der Struktur maximiert werden sollte, konnten von der RLS-Heuristik sinnvoll unterstützt werden. Es ist auch denkbar, die RLS-Heuristik in Optimierungsaufgaben zu nutzen, die eine gewünschte Steifigkeit bei minimaler Beschleunigung fordern. In dieser Dissertation konnte für ein konkretes Beispiel gezeigt werden, dass dies zwar theoretisch möglich ist, praktisch aber wenig Nutzen aus der Anwendung der RLS-Heuristik extrahiert werden konnte. Tabelle 8–1 fasst die Optimierungsergebnisse kompakt zusammen.

Tabelle 8–1: Zusammenfassung der Zielfunktionswerte der praktischen Optimierungsaufgaben (*Lockere Fertigungsrestriktionen* (LFR), *Praxisnahe Fertigungsrestriktionen* (PFR))

Optimierungsmodell	Zielfunktion (min)	Zielfunktionswerte		
		Initial	Opt. mit RLS	Opt. ohne RLS
Rahmenmodell	$u_{y,\max}$ LFR	72,36 mm	7,12 mm	11,80 mm
	$u_{y,\max}$ PFR	72,36 mm	7,82 mm	14,28 mm
	$u_{y,\max}$ PFR + Formopt.	12,01 mm	7,75 mm	8,27 mm
	$\ddot{u}_{-y,\max}$ LFR + Dim.	4661,42 m/s^2	1302,34 m/s^2	1289,51 m/s^2
Biegeträger	$u_{-z,\max}$ PFR	60,70 mm	30,02 mm	34,02 mm
Schwellerausschnitt	$u_{-y,\max}$ PFR	69,00 mm	33,04 mm	31,22 mm

Insbesondere die Optimierung am Biegeträger hat gezeigt, dass die Heuristik auch auf einem völlig fremden Modell und Lastfall einen Mehrwert bezüglich des gefundenen optimierten Entwurfs bieten kann. Aber auch bei der Optimierung des Schwellerausschnitts konnte die RLS-Heuristik nachvollziehbare Beiträge zur Optimierung beisteuern.

Welche Grenzen hat die umgesetzte RL-Methodik?

Eine große Herausforderung für die RLS-Heuristik ist die Kopplung zwischen der Zelle und dem gesamten strukturmechanischen Verhalten, welches in der GHT-Optimierung von eigentlichem Interesse ist. Insbesondere in späteren Iterationen der Optimierung sind die Zellen naturgemäß klein und deren Einfluss auf das gesamte Strukturverhalten ist gering. Mit dem geschätzten Einfluss auf das Strukturverhalten wurde versucht die Zellen entsprechend sinnvoll auszuwählen, aber auch hier ist der Erfolg von den gefundenen Entwürfen in der Optimierung und der zugehörigen Aufgabenstellung abhängig. So schadet die lokale Änderung von kleinen Zellen häufig sogar der Gesamtstruktur, da durch das Einbringen neuer Wände die Wandstärke der Gesamtstruktur kleiner wird, wodurch diese entgegen der betrachteten steifigkeitsmaximierenden Optimierungsaufgabe weicher wird.

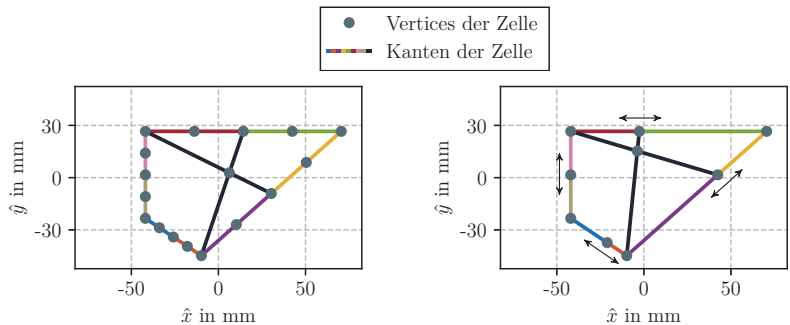
Zusätzlich ist die RLS-Heuristik mit ihrer Möglichkeit zur Anpassung von Form und Topologie aufgrund des Aufbaus der Umgebung an die Rahmenvertices $\partial\mathcal{V}^{\{G_Z\}}$ gebunden. Zwar lassen sich auch so bereits sinnvolle Zellstrukturen finden, allerdings können beispielsweise häufig keine Wände senkrecht zur Impaktorflugrichtung am Auftreffpunkt eingezogen werden, da dort kein Zellenvertex positioniert ist.

In dieser Dissertation wurde auch untersucht, ob die Heuristik dazu genutzt werden kann, weichere anstelle von ausschließlich steiferen Strukturen zu generieren. Hier hat sich gezeigt, dass der lokale Zellansatz dafür nicht gut geeignet ist. Die Zelle wird beim Einbringen von Wänden i. d. R. steifer, weshalb ein modifizierter Ansatz untersucht wurde. Bei diesem modifizierten Ansatz sollten Wände aus Zellen herausgelöscht werden. Dazu wurden Zellen gesucht, welche initial schon mit Wänden nach den Regeln der Trainingsumgebung gefüllt waren. Es zeigt sich allerdings, dass solche mit mehreren Wänden gefüllten Zellen in praktischen GHT-Optimierungen nur äußerst selten vorkommen. Ebenso müsste hier eine weitere Bedingung angeführt werden, damit der Agent nicht einfach alle Kanten aus der Zelle herauslöscht, um eine möglichst weiche Struktur zu generieren.

9 Ausblick

Die in dieser Dissertation beschriebene Methodik der RLS-Heuristik ist in vielerlei Hinsicht modifizier- und erweiterbar. In diesem Kapitel werden zentrale Vorschläge für ein Erweitern der vorgestellten Methodik, aber auch Ideen für neuartige Konzepte vorgeschlagen, welche für weitere Forschung zur Unterstützung der GHT durch Methoden des RLs geeignet sind.

In der vorgestellten Implementierung wird die Form und Topologie von Zellen maßgeblich durch die Verteilung und Anzahl der Vertices entlang des Zellrahmens $\partial\mathcal{E}^{\{G_{\mathrm{Zm}}\}}$ bestimmt. Hier können zwei neue Ansätze zur Modifikation des Kantenteilungsprozesses bei dem Aufbau des Zellengraphen in Erwägung gezogen werden, welche beide in Abbildung 9–1 dargestellt werden.



(a) Teilen der Kanten durch weitere Vertices **(b)** Variation der Vertex-Positionen

Abbildung 9–1: Variationen zur Erweiterung des Kantenteilungsprozesses

Zum einen ist es sinnvoll, wenn der Kantenteilungsprozess so erweitert wird, dass die originalen Zellkanten in mehr als nur zwei Kanten mittig unterteilt werden. Dadurch kann der Agent noch flexibler und situationsgerechter auf das vorliegende Strukturverhalten eingehen. Die Komplexität des Problems nimmt dabei durch einen wachsenden Aktionsraum der Trainingsumgebung aber signifikant zu.

Zum anderen ist es auch möglich, dass der Agent die kantenteilenden Vertices entlang der Kantensegmente verschieben kann. Dadurch erhöht sich die Dimension des Aktionsraums nicht so stark und der Agent kann trotzdem situationsgerechter auf das Strukturverhalten reagieren.

Es ist auch denkbar, sich komplett von dem Zellenansatz zu lösen und einen vollständig graphbasierten Ansatz mit GNNs oder GCNs umzusetzen. Das würde dabei helfen, die Limitierungen des hier vorgestellten Zellansatzes zu umgehen. In Abschnitt 5.3 wurde diskutiert, warum dies in dieser Dissertation nicht umgesetzt wurde. Das bedeutet aber nicht, dass es grundsätzlich unmöglich ist und nicht ausprobiert werden sollte. Insbesondere mit dem Fortschreiten der Entwicklung und der Integration von Graph-Räumen in STABLE-BASELINES3 wird ein solcher Ansatz leichter ermöglicht.

In dieser Dissertation wurde nicht direkt darauf eingegangen, wie es umgesetzt werden kann, dass die Agenten Kanten in gezielter Reihenfolge in die Zelle einbringen. So wäre es sinnvoll, wenn sichergestellt wäre, dass der Agent als erstes die Kante in die Zelle einbringt, die die Strukturperformance am stärksten verbessert. Dann könnte beispielsweise ein weiterer Check in die RLS-Heuristik integriert werden, welcher die RLS-Heuristik frühzeitig beendet, weil die gewünschte Performanceverbesserung nicht eingetreten ist. So können dann unnötige Funktionsaufrufe vermieden werden. Ein Agent mit myopischem Verhalten würde zwar die Aktion mit dem bestmöglichen Reward für den aktuellen Zustand anstreben, allerdings wird dadurch verhindert, dass der Agent bessere Lösungen durch das Vorhersehen über mehrere Schritte hinweg findet. Hier sollten dann weiter ausgearbeitete Ansätze entwickelt werden.

Die Betrachtung einer zur Maximierung der Steifigkeit gegensätzlichen Optimierungsaufgabe wurde in dieser Dissertation diskutiert und mit dem Zellenansatz als schwer lösbar identifiziert. Nichtsdestotrotz sollte in diesem Bereich weiter geforscht werden, um die GHT in einem möglichst breiten Spektrum unterstützen zu können. Dabei ist es auch denkbar, dass ein abgewandelter oder ganz anderer Ansatz als der Zellansatz zum Tragen kommt.

Eine weitere sinnvolle Erweiterungen der Methode ist das Trainieren der Agenten auf diverseren Simulationsmodellen mit verschiedenen Impaktoren und vor allem unterschiedlichen Materialmodellen mit Versagenskriterien, wie beispielsweise Stahl oder Faserverbundwerkstoffen. Mit der Betrachtung von Stahl und Faserverbundwerkstoffen kann die RLS-Heuristik auch für die Erweiterungen der GHT mit Blechstrukturen und gewickelten Kompositstrukturen eingesetzt werden.

Final kann die in dieser Dissertation vorgestellte Methode auch auf die GHT-3D übertragen werden, wobei Zellen dann durch einen räumlich dreidimensionalen Graphen repräsentiert würden. Das begünstigt das Finden von steiferen Anordnungen von Profilen in Profilstrukturen.

Vorveröffentlichungen zu dieser Dissertation

Trilling J, Schumacher A, Zhou M (2022a) *Einsatz von Reinforcement Learning zur lokalen Versteifung von Extrusionsprofilen in Crashlastfällen*. NAFEMS Online-Magazin 62: 59–70 (siehe S. 75, 76, 78, 79, 112)

Trilling J, Schumacher A, Zhou M (2022b) *Generation of Designs for Local Stiffness Increase of Crash Loaded Extrusion Profiles with Reinforcement Learning*. Machine Learning and Artificial Intelligence in CFD and Structural Analysis. NAFEMS. Wiesbaden (siehe S. 78)

Trilling J, Schumacher A, Zhou M (2024) *Reinforcement learning based agents for improving layouts of automotive crash structures*. Applied Intelligence 54: 1751–1769 (siehe S. 70, 73, 74)

Literaturverzeichnis

Bendsøe MP, Kikuchi N (1988) *Generating optimal topologies in structural design using a homogenization method*. Computer Methods in Applied Mechanics and Engineering 71: 197–224 (siehe S. 13)

Bendsøe MP, Sigmund O (2004) *Topology optimization: Theory, methods, and applications*. Springer, Berlin und Heidelberg (siehe S. 13)

Beyer F, Schneider D, Schumacher A (2021) *Finding three-dimensional layouts for crashworthiness load cases using the graph and heuristic based topology optimization*. Structural and Multidisciplinary Optimization 63: 59–73 (siehe S. 1, 16, 20, 23, 24)

Bohn B, Garcke J, Griebel M (2016) *A sparse grid based method for generative dimensionality reduction of high-dimensional data*. Journal of Computational Physics 309: 1–17 (siehe S. 61)

Bohn B, Garcke J, Iza-Teran R, Paprotny A, Peherstorfer B, Schepsmeier U, Thole CA (2013) *Analysis of Car Crash Simulation Data with Nonlinear Machine Learning Methods*. Procedia Computer Science 18: 621–630 (siehe S. 61)

Borsotto D, Strickstrock R, Thole CA (2015) *Improving robustness of Chevrolet Silverado with exemplary design adaptations based on identified scatter sources*. 10th European LS-DYNA Conference 2015. Würzburg (siehe S. 64)

Boser BE, Guyon IM, Vapnik VN (1992) *A training algorithm for optimal margin classifiers*. Proceedings of the Fifth Annual Workshop on Computational Learning Theory. Association for Computing Machinery, New York: 144–152 (siehe S. 32)

Breiman L (1996) *Bagging predictors*. Machine Learning 24: 123–140 (siehe S. 32)

Breiman L (2001) *Random Forests*. Machine Learning 45: 5–32 (siehe S. 32, 34)

Breiman L, Friedman J, Stone CJ, Olshen RA (1984) *Classification and Regression Trees*. Taylor & Francis (siehe S. 32)

Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2016) *OpenAI Gym*. `https://github.com/openai/gym`. Zuletzt aufgerufen am 12.11.2023 (siehe S. 76).

Bujny M (2020) *Level Set Topology Optimization for Crashworthiness using Evolutionary Algorithms and Machine Learning*. Diss. Technische Universität München (siehe S. 14, 15, 63)

Bujny M, Aulig N, Olhofer M, Duddeck F (2016) *Hybrid evolutionary approach for level set topology optimization*. 2016 IEEE Congress on Evolutionary Computation (CEC): 5092–5099 (siehe S. 63)

Bujny M, Aulig N, Olhofer M, Duddeck F (2018) *Identification of optimal topologies for crashworthiness with the evolutionary level set method*. International Journal of Crashworthiness 23: 395–416 (siehe S. 14)

CARHS GmbH (2023) *SafetyCompanion 2023*. `https://www.carhs.de/de/safety-companion-overview.html`. Zuletzt aufgerufen am 28.12.2023 (siehe S. 7).

Chandrasekhar A, Suresh K (2021) *TOuNN: Topology Optimization using Neural Networks*. Structural and Multidisciplinary Optimization 63: 1135–1149 (siehe S. 66)

Choi WS, Park GJ (2002) *Structural optimization using equivalent static loads at all time intervals*. Computer Methods in Applied Mechanics and Engineering 191: 2105–2122 (siehe S. 14)

Clemens P, Schumacher A (2023) *Nested loop approach for topology and shape optimization of crash-loaded deep-drawn components using contact forces for the inner loops*. Structures 55: 2013–2022 (siehe S. 15)

Cortes C, Vapnik V (1995) *Support-vector networks*. Machine Learning 20: 273–297 (siehe S. 32)

Cover T, Hart P (1967) *Nearest neighbor pattern classification*. IEEE Transactions on Information Theory 13: 21–27 (siehe S. 32)

Diez C (2018) *qd – Build your own LS-DYNA® Tools Quickly in Python*. 15th International LS-DYNA Users Conference. Detroit (siehe S. 77)

Diez C (2019) *Process for Extraction of Knowledge from Crash Simulations by means of Dimensionality Reduction and Rule Mining*. Diss. Bergische Universität Wuppertal (siehe S. 59, 62)

Diez C, Harzheim L, Schumacher A (2016) *Effiziente Wissensgenerierung zur Robustheitsuntersuchung von Fahrzeugstrukturen mittels Modellreduktion und Ähnlichkeitsanalyse*. VDI-Berichte 2279 (siehe S. 59, 62)

Ernst H, Schmidt J, Beneken GH (2020) *Grundkurs Informatik: Grundlagen und Konzepte für die erfolgreiche IT-Praxis — Eine umfassende, praxisorientierte Einführung*. 7. Auflage. Springer Vieweg, Wiesbaden (siehe S. 32)

Ertel W (2016) *Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung*. 4. Auflage. Springer Vieweg, Wiesbaden (siehe S. 27, 38, 45)

Frochte J (2020) *Maschinelles Lernen: Grundlagen und Algorithmen in Python*. 3. Auflage. Hanser, München (siehe S. 27, 28, 30–34, 42–44)

Gao D, Yao B, Chang G, Li Q (2022) *Multi-Objective Optimization Design of Vehicle Side Crashworthiness Based on Machine Learning Point-Adding Method*. Applied Sciences 12: 10320 (siehe S. 62)

Garcke J, Iza-Teran R (2015) *Machine Learning Approaches for Repositories of Numerical Simulation Results*. 10th European LS-DYNA Conference 2015. Würzburg (siehe S. 61)

Garcke J, Iza-Teran R (2017) *Machine Learning Approaches for Data from Car Crashes and Numerical Car Crash Simulations*. NAFEMS World Congress 2017. NAFEMS. Stockholm (siehe S. 59)

Géron A (2019) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2. Auflage. O'Reilly (siehe S. 29, 30)

Gori M, Monfardini G, Scarselli F (2005) *A new model for learning in graph domains*. 2005 IEEE International Joint Conference on Neural Networks: 729–734 (siehe S. 2, 38)

Haarnoja T, Ha S, Zhou A, Tan J, Tucker G, Levine S (2019) *Learning to Walk via Deep Reinforcement Learning*. Robotics: Science and Systems XV. Freiburg im Breisgau (siehe S. 46)

Hagberg AA, Schult DA, Swart PJ (2008) *Exploring network structure, dynamics, and function using NetworkX*. Proceedings of the 7th Python in Science Conference. Hrsg. von G Varoquaux, T Vaught, J Millman. Pasadena: 11–15 (siehe S. 76, 86)

Hahner S, Iza-Teran R, Garcke J (2020) *Analysis and Prediction of Deforming 3D Shapes Using Oriented Bounding Boxes and LSTM Autoencoders*. Artificial Neural Networks and Machine Learning. Bratislava (siehe S. 58, 62)

Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, Del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE (2020) *Array programming with NumPy*. Nature 585: 357–362 (siehe S. 77)

Hayashi K, Ohsaki M (2020) *Reinforcement Learning and Graph Embedding for Binary Truss Topology Optimization Under Stress and Displacement Constraints*. Frontiers in Built Environment 6 (siehe S. 67)

Hayashi K, Ohsaki M (2022) *Graph-based reinforcement learning for discrete cross-section optimization of planar steel frames*. Advanced Engineering Informatics 51 (siehe S. 67)

Hochreiter S, Schmidhuber J (1997) *Long short-term memory*. Neural Computation 9: 1735–1780 (siehe S. 38)

Huang S, Ontañón S (2022) *A Closer Look at Invalid Action Masking in Policy Gradient Algorithms*. International Florida Artificial Intelligence Research Society Conference Proceedings. Bd. 35. Hutchinson Island, Florida (siehe S. 149)

Humm BG, Buxmann P, Schmidt JC (2022) *Grundlagen und Anwendungen von KI*. Künstliche Intelligenz in der Forschung: Neue Möglichkeiten und Herausforderungen für die Wissenschaft. Springer, Berlin und Heidelberg (siehe S. 27, 28).

Hunkeler S (2014) *Topology Optimisation in Crashworthiness Design via Hybrid Cellular Automata for Thin Walled Structures*. Diss. Queen Mary University of London (siehe S. 13)

Ivarsson N, Wallin M, Tortorelli D (2018) *Topology optimization of finite strain viscoplastic systems under transient loads*. International Journal for Numerical Methods in Engineering 114: 1351–1367 (siehe S. 16)

Iza-Teran R, Garcke J (2019) *A Geometrical Method for Low-Dimensional Representations of Simulations*. Society for Industrial & Applied Mathematics (SIAM) / American Statistical Association (ASA) Journal on Uncertainty Quantification 7: 472–496 (siehe S. 59)

Jeong H, Bai J, Batuwatta-Gamage CP, Rathnayaka C, Zhou Y, Gu Y (2023) *A Physics-Informed Neural Network-based Topology Optimization (PINNTO) framework for structural optimization*. Engineering Structures 278 (siehe S. 66)

Johnson DB (1975) *Finding All the Elementary Circuits of a Directed Graph*. Society for Industrial & Applied Mathematics (SIAM) Journal on Computing 4: 77–84 (siehe S. 86)

Jones DR, Schonlau M, Welch WJ (1998) *Efficient Global Optimization of Expensive Black-Box Functions*. Journal of Global Optimization 13: 455–492 (siehe S. 63)

Kazi MK, Eljack F, Mahdi E (2022) *Design of composite rectangular tubes for optimum crashworthiness performance via experimental and ANN techniques*. Composite Structures 279 (siehe S. 62)

Kingma DP, Ba J (2015) *Adam: A Method for Stochastic Optimization*. 3rd International Conference on Learning Representations. San Diego (siehe S. 113)

Kipf TN, Welling M (2017) *Semi-Supervised Classification with Graph Convolutional Networks*. 5th International Conference on Learning Representations. Toulon (siehe S. 38)

Klein B (2015) *FEM: Grundlagen und Anwendungen der Finite-Element-Methode im Maschinen- und Fahrzeugbau*. 10. Auflage. Springer Vieweg, Wiesbaden (siehe S. 8)

Koch M, Bäck T (2018) *Machine Learning for Predicting the Impact Point of a Low Speed Vehicle Crash*. 17th IEEE International Conference on Machine Learning and Applications. Orlando: 1432–1437 (siehe S. 60)

Koch M, Wang H, Bäck T (2018) *Machine Learning for Predicting the Damaged Parts of a Low Speed Vehicle Crash*. Thirteenth International Conference on Digital Information Management. Berlin: 179–184 (siehe S. 60)

Kohar CP, Greve L, Eller TK, Connolly DS, Inal K (2021) *A machine learning framework for accelerating the design process using CAE simulations: An application to finite element analysis in structural crashworthiness*. Computer Methods in Applied Mechanics and Engineering 385: 114008 (siehe S. 58, 59)

Konda VR, Tsitsiklis JN (1999) *Actor-Critic Algorithms*. Advances in Neural Information Processing Systems 12 (siehe S. 54)

Kracker D, Dhanasekaran RK, Schumacher A, Garcke J (2023) *Method for automated detection of outliers in crash simulations*. International Journal of Crashworthiness 28: 96–107 (siehe S. 61)

Kracker D, Garcke J, Schumacher A (2020) *Automatic Analysis of Crash Simulations with Dimensionality Reduction Algorithms such as PCA and t-SNE*. 16th International LS-DYNA Users Conference (siehe S. 58, 61)

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) *Gradient-based learning applied to document recognition*. Proceedings of the IEEE 86: 2278–2324 (siehe S. 31, 42)

Lei X, Liu C, Du Z, Zhang W, Guo X (2019) *Machine Learning-Driven Real-Time Topology Optimization Under Moving Morphable Component-Based Framework*. Journal of Applied Mechanics 86 (siehe S. 66)

Li Y, Ni P, Chang V (2020) *Application of deep reinforcement learning in stock trading strategies and stock forecasting*. Computing 102: 1305–1322 (siehe S. 46)

Li Z, Ma W, Yao S, Xu P, Hou L, Deng G (2021) *A machine learning based optimization method towards removing undesired deformation of energy-absorbing structures*. Structural and Multidisciplinary Optimization 64: 919–934 (siehe S. 57, 60)

Link S, Schneider D, Schumacher A, Ortmann C (2018) *Integration of Flange Connections in the Graph and Heuristic Based Topology Optimization of Crashworthiness Structures*. EngOpt 2018 Proceedings of the 6th International Conference on Engineering Optimization: 619–631 (siehe S. 1, 16)

Liu K, Detwiler D, Tovar A (2017) *Optimal Design of Nonlinear Multimaterial Structures for Crashworthiness Using Cluster Analysis*. Journal of Mechanical Design 139 (siehe S. 63)

Liu K, Detwiler D, Tovar A (2018) *Metamodel-Based Global Optimization of Vehicle Structures for Crashworthiness Supported by Clustering Methods*. Hrsg. von A Schumacher, T Vietor, S Fiebig, KU Bletzinger, K Maute. Advances in Structural and Multidisciplinary Optimization: 1545–1557. Springer, Cham (siehe S. 63)

Livermore Software Technology Corporation (LSTC) (2023a) *Ls-Dyna Manuals*. `https://www.dynasupport.com/manuals/ls-dyna-manuals`. Zuletzt aufgerufen am 06.10.2023 (siehe S. 76).

Livermore Software Technology Corporation (LSTC) (2023b) *Ls-Opt Manuals*. `https://www.lsoptsupport.com/documents/manuals/ls-opt`. Zuletzt aufgerufen am 31.12.2023 (siehe S. 25).

MacQueen J (1967) *Some methods for classification and analysis of multivariate observations*. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability: 281–298. University of California Press (siehe S. 33)

Mayer RR, Kikuchi N, Scott RA (1996) *Application of Topological Optimization Techniques to Structural Crashworthiness*. International Journal for Numerical Methods in Engineering 39: 1383–1403 (siehe S. 13, 14)

McCarthy J, Minsky ML, Rochester M, Shannon CE (1955) *A proposal for the Dartmouth summer research project on artificial intelligence*. `http://raysolomonoff.com/dartmouth/boxa/dart564props.pdf`. Zuletzt aufgerufen am: 06.04.2023 (siehe S. 27).

McCulloch WS, Pitts W (1943) *A logical calculus of the ideas immanent in nervous activity*. The bulletin of mathematical biophysics 5: 115–133 (siehe S. 32, 38)

McKay MD, Beckman RJ, Conover WJ (1979) *A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code*. Technometrics 21: 239 (siehe S. 29)

Mertler SM (2022) *Comparative analysis of crash simulation results using generative nonlinear dimensionality reduction*. Diss. Bergische Universität Wuppertal (siehe S. 64)

Mitchell TM (1997) *Machine Learning*. McGraw-Hill, New York (siehe S. 29)

Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) *Playing Atari with Deep Reinforcement Learning*. Neural Information Processing Systems (NIPS) Deep Learning Workshop. Lake Tahoe (siehe S. 46, 108, 115)

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) *Human-level control through deep reinforcement learning*. Nature 518: 529–533 (siehe S. 108, 115)

Murphy K (2012) *Machine Learning – A Probabilistic Perspective*. Adaptive Computation and Machine Learning. MIT Press, Cambridge (siehe S. 117)

Narayanan A, Chandramohan M, Venkatesan R, Chen L, Liu Y, Jaiswal S (2017) *graph2vec: Learning Distributed Representations of Graphs* (siehe S. 69)

Olschinka C, Schumacher A (2008) *Graph Based Topology Optimization of Crashworthiness Structures*. Proceedings in Applied Mathematics and Mechanics 8 (siehe S. 1, 16)

Ortmann C, Schumacher A (2013) *Graph and heuristic based topology optimization of crash loaded structures*. Structural and Multidisciplinary Optimization 47: 839–854 (siehe S. 1, 16, 72)

Ortmann C (2015) *Entwicklung eines graphen- und heuristikbasierten Verfahrens zur Topologieoptimierung von Profilquerschnitten für Crashlastfälle*. Diss. Bergische Universität Wuppertal (siehe S. 1, 16, 20, 72, 83, 178)

Ortmann C, Sperber J, Schneider D, Link S, Schumacher A (2021) *Crashworthiness design of cross-sections with the Graph and Heuristic based Topology Optimization incorporating competing designs*. Structural and Multidisciplinary Optimization 64: 1063–1077 (siehe S. 1, 16, 20–23)

Park GJ (2011) *Technical overview of the equivalent static loads method for non-linear static response structural optimization*. Structural and Multidisciplinary Optimization 43: 319–337 (siehe S. 14)

Pascanu R, Mikolov T, Bengio Y (2012) *On the difficulty of training Recurrent Neural Networks*. `https://arxiv.org/pdf/1211.5063`. Zuletzt aufgerufen am 13.11.2023 (siehe S. 112).

Patel NM, Kang BS, Renaud JE, Tovar A (2009) *Crashworthiness Design Using Topology Optimization*. Journal of Mechanical Design 131 (siehe S. 13)

Patel NM (2007) *Crashworthiness Design using Topology Optimization*. Diss. University Of Notre Dame (siehe S. 13, 14)

Pearson K (1901) *On lines and planes of closest fit to systems of points in space*. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2: 559–572 (siehe S. 34)

Pedersen CBW (2003) *Topology optimization design of crushed 2D-frames for desired energy absorption history*. Structural and Multidisciplinary Optimization 25: 368–382 (siehe S. 15)

Pedersen CBW (2004) *Crashworthiness design of transient frame structures using topology optimization*. Computer Methods in Applied Mechanics and Engineering 193: 653–678 (siehe S. 15)

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research 12: 2825–2830 (siehe S. 37)

Quinlan JR (1986) *Induction of decision trees*. Machine Learning 1: 81–106 (siehe S. 32)

Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N (2021) *Stable-Baselines3: Reliable Reinforcement Learning Implementations*. Journal of Machine Learning Research 22: 1–8 (siehe S. 76)

Rich E (1983) *Artificial Intelligence*. McGraw-Hill, New York (siehe S. 27)

Robbins H, Monro S (1951) *A Stochastic Approximation Method*. The Annals of Mathematical Statistics 22: 400–407 (siehe S. 32, 41)

Rosenblatt F (1958) *The perceptron: a probabilistic model for information storage and organization in the brain*. Psychological Review 65: 386–408 (siehe S. 32, 38)

Rozvany GIN, Zhou M, Birker T (1992) *Generalized shape optimization without homogenization*. Structural and Multidisciplinary Optimization 4: 250–252 (siehe S. 13)

Rumelhart DE, Hinton GE, Williams RJ (1986) *Learning representations by back-propagating errors*. Nature 323: 533–536 (siehe S. 32, 41, 42)

Samuel AL (1959) *Some Studies in Machine Learning Using the Game of Checkers*. IBM Journal of Research and Development 3: 210–229 (siehe S. 29)

Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2009) *The graph neural network model*. IEEE Transactions on Neural Networks 20: 61–80 (siehe S. 2, 38)

Schneider D (2023) *Graphen- und heuristikbasierte Topologieoptimierung von Profilstrukturen aus Faser-Kunststoff-Verbunden in Crashanwendungen*. Diss. Bergische Universität Wuppertal (siehe S. 16, 21, 172)

Schneider D, Schumacher A, Donhauser T, Huf A, Schmeer S (2019) *Flexible Graph Syntax for the Topology Optimization of Crashworthiness Profile Structures Made from Thermoplastic Composites*. Key Engineering Materials 809: 493–499 (siehe S. 1, 16)

Schulman J, Moritz P, Levine S, Jordan M, Abbeel P (2015) *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. `https://arxiv.org/pdf/1506.02438`. Zuletzt aufgerufen am 13.11.2023 (siehe S. 109).

Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) *Proximal Policy Optimization Algorithms*. `https://arxiv.org/pdf/1707.06347`. Zuletzt aufgerufen am 13.11.2023 (siehe S. 109, 110).

Schumacher A (2020) *Optimierung mechanischer Strukturen: Grundlagen und industrielle Anwendungen*. 3. Auflage. Springer Vieweg, Berlin (siehe S. 9–15, 33)

Schumacher A, Ortmann C (2013) *Rule generation for optimal topology changes of crash-loaded structures*. 10th World Congress on Structural and Multidisciplinary Optimization. Orlando (siehe S. 20)

Schwanitz P (2016) *Robuste Optimierung und Bewertung von parametrisch modellierten Crashboxen*. Diss. Technische Universität Berlin (siehe S. 6)

Shannon CE (1948) *A Mathematical Theory of Communication*. Bell System Technical Journal 27: 379–423 (siehe S. 32)

SIDACT GmbH (2024) *DIFFCRASH: Improve Robustness Of Crash Simulation Models*. `https://www.sidact.de/diffcrash`. Zuletzt aufgerufen am 03.01.2024 (siehe S. 64).

Society of Automotive Engineers (1995) *SAE J211-1: Instrumentation for Impact Test*. `https://law.resource.org/pub/us/cfr/ibr/005/sae.j211-1.1995.pdf`. Zuletzt aufgerufen am 05.07.2023 (siehe S. 105).

Sosnovik I, Oseledets I (2017) *Neural networks for topology optimization*. `https://arxiv.org/pdf/1709.09578`. Zuletzt aufgerufen am 13.11.2023 (siehe S. 66).

Spearman C (1904) *The Proof and Measurement of Association between Two Things*. The American Journal of Psychology 15: 72 (siehe S. 35)

Sperber J (2022) *Graphen- und Heuristikbasierte Topologieoptimierung axial belasteter Crashstrukturen*. Diss. Bergische Universität Wuppertal (siehe S. 14, 16)

Stander N, Craig KJ (2002) *On the robustness of a simple domain reduction scheme for simulation–based optimization*. Engineering Computations 19: 431–450 (siehe S. 25)

Stanley KO, Miikkulainen R (2002) *Evolving Neural Networks through Augmenting Topologies*. Evolutionary Computation 10: 99–127 (siehe S. 2, 67)

Stone M (1974) *Cross-Validatory Choice and Assessment of Statistical Predictions*. Journal of the Royal Statistical Society. Series B (Methodological) 36: 111–147 (siehe S. 36)

Sutton RS, Barto AG (2015) *Reinforcement Learning: An Introduction*. Cambridge (siehe S. 55).

Sutton RS, Barto AG (2020) *Reinforcement Learning: An Introduction*. `http://incompleteideas.net/book/RLbook2020.pdf`. Zuletzt aufgerufen am 13.11.2023. Cambridge (siehe S. 44, 45, 52).

Taylor ME, Whiteson S, Stone P (2006) *Comparing evolutionary and temporal difference methods in a reinforcement learning domain*. Proceedings of the 8th annual conference on Genetic and evolutionary computation. Association for Computing Machinery, New York: 1321–1328 (siehe S. 68)

Thole CA, Nikitina L, Nikitin I, Clees T (2010) *Advanced Mode Analysis for Crash Simulation Results*. 9. LS-DYNA Forum. Bamberg (siehe S. 64)

Triller J, Immel R, Harzheim L (2022) *Difference-based Equivalent Static Load Method with adaptive time selection and local stiffness adaption*. Structural and Multidisciplinary Optimization 65: 1–17 (siehe S. 105)

Triller J, Immel R, Timmer A, Harzheim L (2021) *The difference-based equivalent static load method: an improvement of the ESL method's nonlinear approximation quality*. Structural and Multidisciplinary Optimization 63: 2705–2720 (siehe S. 14)

Trilling J, Schumacher A, Zhou M (2022a) *Einsatz von Reinforcement Learning zur lokalen Versteifung von Extrusionsprofilen in Crashlastfällen*. NAFEMS Online-Magazin 62: 59–70 (siehe S. 75, 76, 78, 79, 112)

Trilling J, Schumacher A, Zhou M (2022b) *Generation of Designs for Local Stiffness Increase of Crash Loaded Extrusion Profiles with Reinforcement Learning*. Machine Learning and Artificial Intelligence in CFD and Structural Analysis. NAFEMS. Wiesbaden (siehe S. 78)

Trilling J, Schumacher A, Zhou M (2024) *Reinforcement learning based agents for improving layouts of automotive crash structures*. Applied Intelligence 54: 1751–1769 (siehe S. 70, 73, 74)

van der Maaten L, Hinton G (2008) *Visualizing Data using t-SNE*. Journal of Machine Learning Research 9: 2579–2605 (siehe S. 34)

Wagner M (2019) *Lineare und nichtlineare FEM: Eine Einführung mit Anwendungen in der Umformsimulation mit LS-DYNA®*. 2. Auflage. Springer Vieweg, Wiesbaden und Heidelberg (siehe S. 8)

Wang J, Jing F, He M (2023) *Stock Trading Strategy of Reinforcement Learning Driven by Turning Point Classification*. Neural Processing Letters 55: 3489–3508 (siehe S. 46)

Weider K (2021) *Topologische Ableitung zur Optimierung crashbelasteter Strukturen*. Diss. Bergische Universität Wuppertal (siehe S. 15)

Welford BP (1962) *Note on a Method for Calculating Corrected Sums of Squares and Products*. Technometrics 4: 419–420 (siehe S. 114)

Wolfram S (2002) *A new kind of science*. Wolfram Media (siehe S. 13)

Wriggers P (2001) *Nichtlineare Finite-Element-Methoden*. Springer, Berlin und Heidelberg (siehe S. 8, 9)

Wright S (1921) *Correlation and Causation*. Journal of Agricultural Research 20: 557–585 (siehe S. 34)

Zeng D (2019) *Enhanced hybrid cellular automata method for crashworthiness topology optimization of thin-walled structures*. Diss. Technische Universität München (siehe S. 13)

Zhang Y, Peng B, Zhou X, Xiang C, Wang D (2019) *A deep Convolutional Neural Network for topology optimization with strong generalization ability*. `https://arxiv.org/pdf/1901.07761`. Zuletzt aufgerufen am 13.11.2023 (siehe S. 66, 68).

Anhang A

Trainingshistorien der Hyperparameteruntersuchung

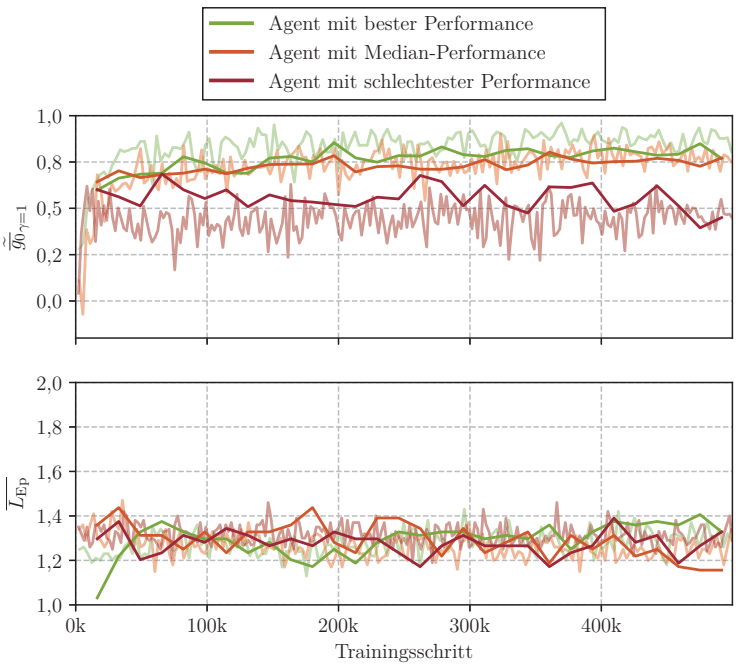


Abbildung A–1: Trainingshistorien der Hyperparameteruntersuchung für 3er-Zellen-Agenten auf dem Evaluationsdatensatz. Die zugehörigen Rollout-Historien werden in gleicher Farbe semi-transparent dargestellt

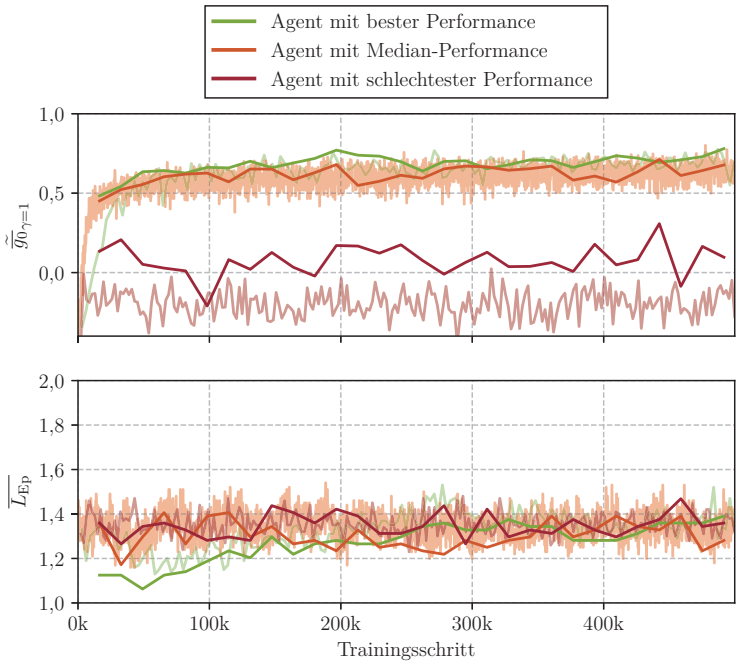


Abbildung A–2: Trainingshistorien der Hyperparameteruntersuchung für 5er-Zellen-Agenten auf dem Evaluationsdatensatz. Die zugehörigen Rollout-Historien werden in gleicher Farbe semi-transparent dargestellt

Anhang B

Tabellarische Zusammenfassungen der praktischen Optimierungen

B.1 Rahmenmodell

B.1.1 Minimierung der Impaktorverschiebung

Lockere Fertigungsrestriktionen

Tabelle B–1: Zusammenfassung der Optimierung der Impaktorverschiebung $u_{y,\max}$ bei dem Rahmenmodell mit *aktiver* RLS-Heuristik unter lockeren Fertigungsrestriktionen

Iteration	Heuristik	$u_{y,\max}$	n_{sim} [davon durch RLS]	
			implementiert	theoretisch
0	Initiale Berechnung	72,36 mm	1 [0]	1 [0]
1	RLS	31,80 mm	8 [4]	6 [2]
2	UDST	20,07 mm	42 [16]	32 [6]
3	DNW	11,20 mm	42 [13]	33 [4]
4	DNW	8,08 mm	43 [16]	33 [6]
5	UDSC	7,75 mm	42 [15]	32 [5]
6	BED	7,12 mm	41 [16]	31 [6]
7	/	/	29 [12]	20 [3]
8	/	/	29 [9]	22 [2]
9	Finale Berechnung	7,12 mm	1 [0]	1 [0]
		Σ	278 [101]	211 [34]

Tabelle B–2: Zusammenfassung der Optimierung der Impaktorverschiebung $u_{y,\max}$ bei dem Rahmenmodell mit *inaktiver* RLS-Heuristik unter lockeren Fertigungsrestriktionen

Iteration	Heuristik	$u_{y,\max}$	n_{sim}
0	Initiale Berechnung	72,36 mm	1
1	BED	54,79 mm	4
2	UDST	29,54 mm	20
3	UDST	30,07 mm	27
4	UDSC	28,82 mm	28
5	BED	16,28 mm	26
6	UDST	16,80 mm	27
7	DNW	15,74 mm	24
8	UDST	15,68 mm	19
9	UDST	13,96 mm	19
10	UDSC	12,92 mm	25
11	BED	12,00 mm	25
12	BED	12,13 mm	22
13	DNW	11,80 mm	24
14	/	/	24
15	/	/	20
16	Finale Berechnung	11,80 mm	1
		Σ	336

Praxisnahe Fertigungsrestriktionen

Tabelle B–3: Zusammenfassung der Optimierung der Impaktorverschiebung $u_{y,\max}$ bei dem Rahmenmodell mit *aktiver* RLS-Heuristik unter praxisnahen Fertigungsrestriktionen

Iteration	Heuristik	$u_{y,\max}$	n_{sim} [davon durch RLS]	
			implementiert	theoretisch
0	Initiale Berechnung	72,36 mm	1 [0]	1 [0]
1	RLS	31,80 mm	8 [4]	6 [2]
2	DNW	27,21 mm	39 [13]	31 [5]
3	DNW	11,71 mm	43 [15]	33 [5]
4	UDST	7,82 mm	43 [15]	33 [5]
5	/	/	35 [15]	25 [5]
6	/	/	27 [7]	21 [1]
7	Finale Berechnung	7,82 mm	1 [0]	1 [0]
		Σ	197 [69]	151 [23]

Tabelle B–4: Zusammenfassung der Optimierung der Impaktorverschiebung $u_{y,\max}$ bei dem Rahmenmodell mit *inaktiver* RLS-Heuristik unter praxisnahen Fertigungsrestriktionen

Iteration	Heuristik	$u_{y,\max}$	n_{sim}
0	Initiale Berechnung	72,36 mm	1
1	BED	54,79 mm	4
2	UDST	35,60 mm	20
3	BED	28,17 mm	29
4	BED	19,88 mm	29
5	SBW	18,48 mm	30
6	UDST	17,73 mm	27
7	SBW	15,56 mm	26
8	BED	14,51 mm	26
9	UDST	14,28 mm	22
10	/	/	28
11	/	/	9
12	Finale Berechnung	14,28 mm	1
		Σ	252

Aktive Formoptimierung

Tabelle B–5: Zusammenfassung der Optimierung der Impaktorverschiebung $u_{y,\max}$ bei dem Rahmenmodell mit *aktiver* RLS-Heuristik und Formoptimierung

Iteration	Heuristik	$u_{y,\max}$	n_{sim} [davon durch RLS]	
			implementiert	theoretisch
0	SSO	12,01 mm	49 [0]	49 [0]
1	RLS	9,88 mm	7 [3]	5 [1]
2	UDST	7,75 mm	35 [11]	29 [5]
3	/	/	27 [9]	20 [2]
4	/	/	24 [5]	20 [1]
5	SSO	7,75 mm	181 [0]	181 [0]
		Σ	323 [28]	304 [9]

Tabelle B–6: Zusammenfassung der Optimierungsergebnisse des Rahmenmodells mit *inaktiver* RLS-Heuristik unter praxisnahen Fertigungsrestriktionen und Formoptimierung

Iteration	Heuristik	$u_{y,\max}$	n_{sim}
0	SSO	12,01 mm	49
1	SBW	8,39 mm	4
2	SLE	8,35 mm	19
3	/	/	22
4	/	/	17
5	SSO	8,27 mm	199
		Σ	310

B.1.2 Minimierung der maximalen Impaktorbeschleunigung

Tabelle B–7: Zusammenfassung der Optimierungsergebnisse des Rahmenmodells mit *aktiver* RLS-Heuristik unter lockeren Fertigungsrestriktionen. Die funktionale Restriktion $\dot{u}_{y,\mathrm{end}} \leq 0\,\mathrm{mm/s}$ ist für alle gezeigten Entwürfe erfüllt

Iteration	Heuristik	$\ddot{u}_{-y,\max}$	$u_{y,\max}$	n_{sim} [davon durch RLS]	
				implementiert	theoretisch
0	SSO	$4661{,}42\,\mathrm{m/s^2}$	$48{,}41\,\mathrm{mm}$	10 [0]	10 [0]
1	UDST	$2432{,}23\,\mathrm{m/s^2}$	$48{,}17\,\mathrm{mm}$	84 [19]	82 [17]
2	SLE	$1465{,}79\,\mathrm{m/s^2}$	$49{,}99\,\mathrm{mm}$	407 [76]	397 [66]
3	RLS	$1302{,}34\,\mathrm{m/s^2}$	$49{,}96\,\mathrm{mm}$	509 [70]	499 [60]
4	/	/	/	466 [82]	456 [72]
5	/	/	/	480 [70]	470 [60]
6	SSO	$1302{,}34\,\mathrm{m/s^2}$	$49{,}96\,\mathrm{mm}$	10 [0]	10 [0]
			Σ	1966 [317]	1924 [275]

Tabelle B–8: Zusammenfassung der Optimierungsergebnisse des Rahmenmodells mit *inaktiver* RLS-Heuristik unter lockeren Fertigungsrestriktionen. Die funktionale Restriktion $\dot{u}_{y,\mathrm{end}} \leq 0\,\mathrm{mm/s}$ ist für alle gezeigten Entwürfe erfüllt

Iteration	Heuristik	$\ddot{u}_{-y,\max}$	$u_{y,\max}$	n_{sim}
0	SSO	$4661{,}42\,\mathrm{m/s^2}$	$48{,}41\,\mathrm{mm}$	10
1	UDST	$2432{,}23\,\mathrm{m/s^2}$	$48{,}17\,\mathrm{mm}$	65
2	SBW	$1801{,}70\,\mathrm{m/s^2}$	$48{,}82\,\mathrm{mm}$	330
3	SLE	$1434{,}73\,\mathrm{m/s^2}$	$49{,}99\,\mathrm{mm}$	485
4	BED	$1289{,}51\,\mathrm{m/s^2}$	$49{,}99\,\mathrm{mm}$	369
5	/	/	/	379
6	/	/	/	397
7	SSO	$1289{,}51\,\mathrm{m/s^2}$	$49{,}99\,\mathrm{mm}$	10
			Σ	2045

B.2 Biegeträger

Tabelle B–9: Zusammenfassung der Optimierungsergebnisse des Biegeträgers mit *aktiver* RLS-Heuristik

Iteration	Heuristik	$u_{-z,\max}$	n_{sim} [davon durch RLS]	
			implementiert	theoretisch
0	Initiale Berechnung	60,70 mm	1 [0]	1 [0]
1	SLE	42,24 mm	5 [0]	5 [0]
2	UDST	36,50 mm	30 [7]	25 [2]
3	UDSC	34,14 mm	36 [10]	29 [3]
4	RLS	36,38 mm	28 [4]	25 [1]
5	UDST	33,67 mm	34 [8]	30 [4]
6	SLE	32,45 mm	27 [0]	27 [0]
7	UDSC	33,27 mm	22 [0]	22 [0]
8	DNW	32,73 mm	20 [0]	20 [0]
9	UDSC	31,07 mm	23 [0]	23 [0]
10	DNW	30,02 mm	21 [0]	21 [0]
11	/	/	26 [0]	26 [0]
12	/	/	19 [0]	19 [0]
13	Finale Berechnung	30,02 mm	1 [0]	1 [0]
		Σ	293 [29]	274 [10]

Tabelle B–10: Zusammenfassung der Optimierungsergebnisse des Biegeträgers mit *inaktiver* RLS-Heuristik

Iteration	Heuristik	$u_{-z,\max}$	n_{sim}
0	Initiale Berechnung	60,70 mm	1
1	SLE	42,21 mm	5
2	UDSC	36,17 mm	23
3	UDSC	35,83 mm	26
4	UDSC	35,04 mm	24
5	DNW	34,05 mm	24
6	/	/	23
7	Finale Berechnung	34,02 mm	1
		Σ	127

B.3 Schwellerausschnitt

Tabelle B–11: Zusammenfassung der Optimierungsergebnisse des Schwellerausschnitts mit *aktiver* RLS-Heuristik

Iteration	Heuristik	$u_{-y,\max}$	n_{sim} [davon durch RLS]	
			implementiert	theoretisch
0	Initiale Berechnung	69,00 mm	1 [0]	1 [0]
1	UDSC	49,48 mm	8 [3]	5 [0]
2	RLS	40,69 mm	37 [13]	30 [6]
3	UDSC	36,46 mm	40 [14]	30 [4]
4	UDST	34,47 mm	40 [15]	32 [7]
5	UDST	33,04 mm	42 [16]	34 [8]
6	/	/	28 [0]	28 [0]
7	/	/	28 [0]	28 [0]
8	Finale Berechnung	33,04 mm	1 [0]	1 [0]
		Σ	225 [61]	189 [25]

Tabelle B–12: Zusammenfassung der Optimierungsergebnisse des Schwellerausschnitts mit *inaktiver* RLS-Heuristik

Iteration	Heuristik	$u_{-y,\max}$	n_{sim}
0	Initiale Berechnung	69,00 mm	1
1	UDSC	49,48 mm	5
2	UDST	39,56 mm	24
3	UDST	39,32 mm	25
4	SLE	36,31 mm	26
5	UDST	33,12 mm	28
6	UDST	34,34 mm	29
7	DNW	32,69 mm	27
8	UDSC	31,80 mm	29
9	BED	31,22 mm	27
10	/	/	30
11	/	/	29
12	Finale Berechnung	31,22 mm	1
		Σ	281

Lebenslauf

Persönliche Daten

Jens Trilling

Geburtsdatum	24.10.1994
Geburtsort	Wuppertal

Schulausbildung

2001 bis 2005	Grundschule Dönberg, Wuppertal
2005 bis 2010	Gymnasium Sedanstraße, Wuppertal
2010 bis 2013	Carl-Duisberg-Gymnasium, Wuppertal

Studium

10/2013 bis 09/2018	Maschinenbau *Bergische Universität Wuppertal*, Abschluss: B.Sc.
10/2018 bis 04/2020	Maschinenbau *Bergische Universität Wuppertal*, Abschluss: M.Sc.

Praktikum

01/2018 bis 07/2018	Praktikum und Bachelorarbeit in der *BMW AG*, Fachabteilung Struktureigenschaften

Berufstätigkeit

seit 04/2020	Wissenschaftlicher Mitarbeiter am *Lehrstuhl für Optimierung mechanischer Strukturen* an der *Bergischen Universität Wuppertal*

Dissertationen vom Lehrstuhl für Optimierung mechanischer Strukturen, Fakultät 7, Bergische Universität Wuppertal

1. Dr.-Ing. Christopher Ortmann (2015): *Entwicklung eines graphen- und heuristikbasierten Verfahrens zur Topologieoptimierung von Profilquerschnitten für Crashlastfälle*, Shaker Verlag, ISBN: 978-3-8440-3746-3
2. Dr.-Ing. Robert Dienemann (2018): *Entwicklung einer Optimierungsmethodik für die Form- und Topologieoptimierung von tiefziehbaren Blechstrukturen*, Shaker Verlag, ISBN: 978-3-8440-6196-3
3. Dr.-Ing. Constantin Diez (2018): *Process for Extraction of Knowledge from Crash Simulations by means of Dimensionality Reduction and Rule Mining* [`https://d-nb.info/1182555063/34`]
4. Dr.-Ing. Manuel Ramsaier (2021): *Integration der Topologie- und Formoptimierung in den automatisierten digitalen Entwurf von Fachwerkstrukturen*, Shaker Verlag, ISBN: 978-3-8440-7788-9
5. Dr.-Ing. Niklas Klinke (2021): *Strategien zur Optimierung von flexibel gewalzten Bauteilen in Karosseriestrukturen*, Shaker Verlag, ISBN: 978-3-8440-7936-4
6. Dr.-Ing. Saad Eddine Hafsa (2021): *Topology optimization method for the adaptation of mechanical structures*, Shaker Verlag, ISBN: 978-3-8440-8306-4
7. Dr.-Ing. Katrin Weider (2021): *Topologische Ableitung zur Optimierung crashbelasteter Strukturen*, Shaker Verlag, ISBN: 978-3-8440-8248-7
8. Dr.-Ing. Jana Büttner (2022): *Effiziente Lösungsansätze zur Reduktion des numerischen Ressourcenbedarfs für den operativen Einsatz der Multidisziplinären Optimierung von Fahrzeugstrukturen*, Shaker Verlag, ISBN: 978-3-8440-8560-0
9. Dr.-Ing. Johannes Sperber (2022): *Graphen- und Heuristikbasierte Topologieoptimierung axial belasteter Crashstrukturen*, Shaker Verlag, ISBN: 978-3-8440-8634-8
10. Dr.-Ing. Stefan Mertler (2022): *Comparative Analysis of Crash Simulation Results using Generative Nonlinear Dimensionality Reduction*, Shaker Verlag, ISBN: 978-3-8440-8761-1
11. Dr.-Ing. Sven Wielens (2022): *Automatische Erstellung von Submodellen für die Crashoptimierung von Fahrzeugkarosserien*, Shaker Verlag, ISBN: 978-3-8440-8717-8
12. Dr.-Ing. Dominik Schneider (2023): *Graphen- und heuristikbasierte Topologieoptimierung von Profilstrukturen aus Faser-Kunststoff-Verbunden in Crashanwendungen*, Shaker Verlag, ISBN: 978-3-8440-8884-7
13. Dr.-Ing. David Kracker (2024): *Automatisierte Auswertung von Crashsimulationen unterschiedlicher Fahrzeug-Entwicklungsständen mit Methoden des maschinellen Lernens*, Shaker Verlag, ISBN: 978-3-8440-9424-4

14. Dr.-Ing. Florian Beyer (2024): *Entwicklung eines Optimierungsverfahrens für crashbelastete 3D-Rahmenstrukturen mit der Graphen- und Heuristikbasierten Topologieoptimierung*, Shaker Verlag, ISBN: 978-3-8440-9475-6

15. Dr.-Ing. Michail Schlosser (2024): *Optimierungsverfahren zur Erhöhung der Versagenslasten von Bolzenverbindungen in faserverstärkten Strukturen*, Shaker Verlag, ISBN: 978-3-8440-9411-4