# Modellfreies Lernen optimaler zeitdiskreter Regelungsstrategien für Fertigungsprozesse mit endlichem Zeithorizont

Zur Erlangung des akademischen Grades eines
**Doktors der Ingenieurwissenschaften (Dr.-Ing.)**
von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)

angenommene
**Dissertation**
von

Johannes Dornheim, M.Sc.

| | |
|---|---|
| Tag der mündlichen Prüfung: | 22. Juni 2021 |
| Referent: | Prof. Dr. Peter Gumbsch |
| Korreferent: | Prof. Dr. Norbert Link |

# Kurzfassung

Die Qualität und Leistungsfähigkeit von Bauteilen wird wesentlich von der Ausführung der beteiligten Fertigungsprozesse bestimmt. Das Prozessergebnis hängt – neben dem Anfangszustand des Bauteils und des Prozesses – von dem Prozessverlauf ab. Bei vielen Fertigungsprozessen kann der Prozessverlauf durch zeitlich veränderliche Stellgrößen maßgeblich bestimmt werden. Die Optimierung dieser zeitveränderlichen Größen mit Hinsicht auf die Qualität des Bauteils ist Gegenstand dieser Arbeit. Die Bauteilqualität ergibt sich zum einen aus den makroskopischen Eigenschaften des erzeugten Bauteils und zum anderen aus der Material-Struktur am Ende des Fertigungsprozesses. Beides lässt sich häufig erst im Anschluss an die Prozessausführung, in Form einer Qualitätskontrolle, beurteilen und quantifizieren. *Prozesspfade* sind Sequenzen von Werten der Stellgrößen, die in dieser Arbeit mit Hinsicht auf die Ergebnisqualität optimiert werden. Reale Prozesse sind nicht vollständig determiniert, sondern hängen auch von während des Prozesses schwankenden Prozessbedingungen ab, die häufig nicht direkt messbar sind. Somit können keine allgemein gültigen, optimalen Prozesspfade ermittelt werden. Die Optimierung der Stellgrößen muss vielmehr während der Prozessausführung erfolgen und stellt dann ein Problem der optimalen Regelung dar, wo anstelle der Prozesspfade Regelungsstrategien treten. Diese sind Abbildungen von beobachteten Größen auf Stellgrößen, welche in Hinsicht auf das Prozessergebnis optimiert werden. Herkömmliche Methoden zur optimalen Regelung setzen meist ein Prozessmodell voraus, das gleichzeitig effizient zu berechnen und ausreichend akkurat bezüglich der Aufgabenstellung sein muss. Dies stellt insbesondere bei komplexen nicht-linearen Fertigungsprozessen eine hohe Hürde dar. Ziel der vorliegenden

Arbeit ist deshalb die Entwicklung und Untersuchung von modellfreien Methoden, die selbstständig optimale Regelungsstrategien von Fertigungsprozessen in Hinsicht auf die Ergebnisqualität lernen. Die Basis für derartige Methoden findet sich in Bereichen des bestärkenden maschinellen Lernens und der adaptiven dynamischen Programmierung.

Zur Erreichung dieses übergreifenden Ziels werden in der Arbeit zwei Problemklassen, (a) die Optimierung von Regelungsstrategien partiell beobachtbarer Fertigungsprozesse (bei denen stellvertretend für den Prozesszustand nur einige, davon abhängige Messgrößen vorliegen) unter variierenden Einflüssen und (b) die Struktur-geleitete Optimierung von Fertigungsprozessen (bei denen die Herstellung einer gegebenen Material-Struktur angestrebt wird) definiert und Methoden des bestärkenden Lernens zur Lösung dieser Problemklassen gegenüber dem Stand der Forschung fortentwickelt und untersucht. Dabei werden weitere besondere Aufgabenstellungen in dem Kontext des übergreifenden Ziels, insbesondere die Entscheidungsoptimierung unter sich ändernden Zielvorgaben und die dateneffiziente Entscheidungsoptimierung bei mehreren äquivalenten Zielen, adressiert.

Die entwickelten, generischen Methoden werden für Prozesse der Metallverarbeitung ausgeprägt und in einer virtuellen Surrogat-Umgebung experimentell untersucht. Die physikalische Simulation eines Tiefziehprozesses wird durch Module zur Simulation der variierenden Prozesseinflüsse und der partiellen Beobachtbarkeit erweitert und bildet die Basis der Untersuchungen der Lösungsmethoden für die Problemklasse (a). Die Simulation eines Metall-Bearbeitungsprozesses zur einachsigen Deformation in beliebige Richtungen bildet die Basis der Untersuchungen zur Struktur-geleiteten Optimierung. Die Ergebnisse der Untersuchungen zeigen die Leistungsfähigkeit der entwickelten Methoden im Vergleich zu klassischen Basismethoden. Neben der Leistungsfähigkeit werden die Dateneffizienz und die Robustheit gegenüber Parameterausprägungen der entwickelten Methoden gezeigt und die Auswirkungen einzelner entwickelter Methodenbestandteile auf die Ergebnisse untersucht.

# Abstract

The quality and performance of components depend to a large extent on the execution of the Industrial processes involved in manufacturing. In addition to the initial conditions of the component and the process, the process result depends on the course of the process. In many manufacturing processes, the course of the process can be significantly determined by time-varying manipulated variables. The optimization of these time-dependent quantities with regard to the quality of the component is the subject of this work. The component quality results from the properties of the manufactured component and the achieved material-structure at the end of the manufacturing process. Usually, both can only be assessed and quantified after the execution of the industrial process. *Process paths* are sequences of values of the manipulated variables that are optimized in this thesis with regard to the quality of the process results. The behavior of real processes is not deterministic but depends on process conditions that fluctuate during the process and are often not directly measurable. In this case, no generally valid optimal process path can be determined. The optimization of the manipulated variables must rather take place during the process execution, and instead of process paths, control strategies are optimized. These are mappings of observed variables to manipulated variables, which are optimized with respect to the process result. Conventional methods for such optimal control problems usually require a process model, which must be efficient to compute and at the same time sufficiently accurate with regard to the task at hand. This is a major hurdle, especially when dealing with complex, non-linear manufacturing processes. The aim of the present work is therefore the development and investigation of model-free methods that autonomously

learn optimal control strategies for industrial manufacturing processes with respect to the quality of process results. The basis for such methods can be found in the areas of reinforcement learning and adaptive dynamic programming.
To achieve this overall objective, two problem classes are defined in the thesis: (a) the optimization of control strategies of partially observable industrial manufacturing processes (where, instead of the process state, only a set of dependent measured variables are available) under varying process influences and (b) the structure-guided optimization of industrial manufacturing processes (with the aim of producing prescribed material structures). New reinforcement learning methods are developed and investigated to solve these problem classes. Further special tasks in the context of the overall objective are addressed. In particular, the decision optimization under changing target specifications and the data-efficient decision optimization with multiple equivalent targets.
Developed methods are applied to optimize metalworking processes and are experimentally investigated in a virtual surrogate environment. The physical simulation of a deep-drawing process is expanded by modules for simulating the varying process influences and partial observability as the basis of the investigation of the solution methods for problem class (a). The simulation of a metalworking process for uniaxial deformation in arbitrary directions serves as the basis of the investigations for developed structure-guided optimization methods. The investigation results show the performance of the developed methods compared to classic basic methods. In addition to the performance, the data efficiency and the robustness to parameter expressions of the developed methods are examined and the effects of individual method components on the results are investigated in ablation studies.

# Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als Teil der *Intelligent Systems Research Group* an der Hochschule Karlsruhe in Kooperation mit dem Karlsruher Institut für Technologie. Eine Vielzahl von Personen und Institutionen haben mich auf unterschiedliche Art auf dem Weg hin zur fertigen Arbeit unterstützt.

An erster Stelle gilt mein Dank meinen Referenten. Professor Dr. Peter Gumbsch danke ich für sein Interesse an meiner Arbeit und die Übernahme des Hauptreferats. Professor Dr. Norbert Link danke ich für die Übernahme des Koreferats, sowie die Betreuung und wissenschaftliche Unterstützung meiner Arbeit.

Bei meinen Arbeitskollegen bedanke ich mich für die Zusammenarbeit in einer sehr kameradschaftlichen Atmosphäre und die Durchsicht dieser Arbeit. Besonders danken möchte ich Samuel Zeitvogel, Johannes Wetzel und Tarek Iraki für zahlreiche wertvolle Diskurse. Bei den Kollegiaten und Betreuern des Graduiertenkollegs 1483, insbesondere bei Lukas Morand und Jan Pagenkopf, bedanke ich mich für die fruchtbare interdisziplinäre Zusammenarbeit.

Der Deutschen Forschungsgemeinschaft (DFG), sowie dem Bundesministerium für Forschung und Bildung (BMBF) gilt mein Dank für die Finanzierung meiner Forschungsarbeit.

Zuletzt möchte ich mich herzlich bei meiner Familie und Freunden bedanken, von denen mich jeder Einzelne auf seine Weise unterstützt hat.

Karlsruhe, im Juni 2021 *Johannes Dornheim*

# Inhaltsverzeichnis

# 1 Einleitung

Gegenstand dieser Arbeit ist die Entwicklung und Untersuchung von modellfreien Methoden zum autonomen Lernen von optimalen Regelungsstrategien für Fertigungsprozesse mit endlichem Zeithorizont.
Maschinelle Prozesse zur Fertigung von Bauteilen sind wesentlicher Bestandteil der wirtschaftlichen Wertschöpfung. In Fertigungsprozessen werden Werkstoffe oder Werkstücke im Prozessverlauf in ihrem Zustand so verändert, dass sich am Ende des Prozesses, gewünschte Eigenschaften ergeben. Am Ende einer Ausführung der in der Arbeit behandelten Fertigungsprozesse findet eine automatisierte quantitative Bewertung der Qualität des erzeugten Werkstücks beziehungsweise des erzeugten Materials statt.
Der Prozessverlauf und damit die Qualität des Prozessergebnisses hängt neben einer Menge zeitunabhängiger Prozessparameter auch von zeitabhängigen Stellgrößen des Prozesses ab. Ist der Prozesszustand ausreichend messbar, die Funktionsweise des Prozesses ausreichend bekannt und sind feste, daraus folgende Sollwerte messbarer Größen vorgegeben, können die Stellgrößen durch klassische Regelungsverfahren bestimmt werden. Sollwerte werden dabei im Vorhinein so festgelegt, dass unter anderem die Qualität des Prozessergebnisses sichergestellt ist. Die genannten Bedingungen sind häufig nicht erfüllt. Insbesondere bei komplexen nicht-linearen Fertigungsprozessen sind die Zusammenhänge zwischen Stellgrößen und Ergebnisqualität häufig zu komplex, um Sollwerte im Vorhinein bestimmen zu können. Eine Möglichkeit, mit diesen Fällen umzugehen ist die Definition einer Gütefunktion anstelle der Sollwerte und die Betrachtung der Regelung als Optimierungsproblem.

In der Praxis werden zur Lösung dieser Optimierungsprobleme, der sogenannten optimalen Regelung, üblicherweise Verfahren der *modellprädiktiven Regelung* verwendet, die auf einem Prozessmodell basierend in jedem Prozessschritt ein lokales Optimierungsproblem lösen. Dies erfordert ein Modell des Fertigungsprozesses, das gleichzeitig ausreichend genau das Prozessverhalten wiederspiegelt und ausreichend schnell ist, um die Prozessregelung nicht zu verzögern. Zeitlich veränderliche, äußere Einflüsse auf das Prozessverhalten (wie Werkzeugverschleiß, Material- und Hilfsmittelveränderungen) müssen bekannt sein und bei einer akkuraten Modellbildung berücksichtigt werden, wenn die modellbasierte optimale Regelung diese adaptieren soll. Prozessmodelle die diesen Anforderungen genügen sind für die meisten realen Fertigungsprozesse nicht verfügbar.
In dieser Arbeit werden deshalb Methoden untersucht, die ohne derartige Prozessmodelle auskommen und somit auch in Situationen anwendbar sind, in denen der Aufwand für die Modellierung oder der rechnerische Aufwand für die Anwendung ausreichend akkurater Prozessmodelle in der optimalen Regelung zu hoch ist. Die entwickelten modellfreien Methoden lernen während der Prozessausführung eine optimale Regelungsstrategie für die spezifischen Bedingungen des Fertigungsprozesses.
Als Regelungsstrategie wird in dieser Arbeit eine Abbildung von Zuständen $s$ des Fertigungsprozesses auf Stellgrößen bezeichnet. Der Terminologie des bestärkenden Lernens folgend, werden Stellgrößen im Folgenden als (Regelungs-)Aktionen $a$ bezeichnet. Das Ziel bei der Optimierung der Regelungsstrategie ist die Maximierung einer quantitativen Bewertung der Ergebnis-Qualität. Eine optimale Regelungsstrategie zeichnet sich dadurch aus, dass sie für jeden Zustand $s$ auf diejenige Aktion $a$ abbildet, die in Bezug auf die erwartete Ergebnis-Qualität des Prozesses optimal ist.
In der Arbeit behandelte Fertigungsprozesse erzeugen ein Werkstück oder ein Material in mehreren aufeinanderfolgenden Verarbeitungsschritten mit Regelungsaktionen $a$ und sind als *Markov-Entscheidungsprozess* modelliert. Bei

den betrachteten Prozessen handelt es sich um *Entscheidungsprozesse mit endlichem Zeithorizont*, bei denen der Einfluss der aktuellen Regelungsaktionen auf zukünftige Prozesszustände zeitlich begrenzt ist. Das Verhalten von Fertigungsprozessen hängt neben den Regelungsaktionen häufig von unbekannten, variierenden Prozessbedingungen ab, die nicht oder nur indirekt messbar sind. Darüber hinaus sind Messungen meist mit einer Messunsicherheit versehen. Das Ergebnis einer Aktion, über welche entschieden wird, ist dann nicht präzise vorhersagbar. Der Fertigungsprozess wird in diesem Fall als *partiell-beobachtbarer Markov-Entscheidungsprozess* betrachtet.
Das Lernen von Regelungsstrategien, wie sie in dieser Arbeit behandelt wird, kann damit als Spezialfall hinsichtlich der optimalen Regelung stochastischer Systeme angesehen werden. Im Unterschied zu klassischen Anwendungsfällen der optimalen Regelung sind dabei Gütefunktionswerte nicht in jedem Zeitschritt, sondern lediglich am Ende einer Prozessausführung in Form der Qualitäts-Bewertung gegeben. Im Fall eines deterministischen Prozesses mit gleichbleibendem Startzustand reduziert sich die optimale Regelungsstrategie wieder zu einem optimalen Prozesspfad.
Die in dieser Arbeit entwickelten und untersuchten Verfahren zum autonomen Lernen von Regelungsstrategien sind Methoden des bestärkenden Lernens. Sie lernen in einem interaktiven Lernvorgang aus gezielten Versuchen und den darauf erfolgenden Reaktionen des Prozesses und adaptieren dabei die spezifischen Bedingungen eines Fertigungsprozesses. Entwickelte Methoden sind modellfrei und generisch für zwei unterschiedliche Anwendungsfelder anwendbar, welche die Schwerpunkte dieser Arbeit bilden:

- Optimierung von Regelungsstrategien partiell beobachtbarer Fertigungsprozesse mit endlichem Zeithorizont unter variierenden Prozessbedingungen. Die partielle Beobachtbarkeit bedeutet, dass der Prozesszustand

einer Messung nicht direkt zugänglich ist, sondern lediglich vom Zustand abhängige Messgrößen erfasst werden können. Gleichzeitig wirken nicht erfasste Bedingungen variierend auf den Prozess ein, der in Folge als stochastischer Prozess dargestellt wird.

- Struktur-geleitete Optimierung von Material-Struktur-verändernden Prozessen mit dem Ziel vorgegebene Material-Strukturen zu erreichen. Dabei wird vorausgesetzt, dass das Ziel des Prozesses die Erzeugung bestimmter Material-Strukturen ist, die gewünschte Material-Eigenschaften aufweisen. Hierzu ist die Messbarkeit der Struktur während der Prozessausführung erforderlich, was durch Simulation der Prozesse und – in Ausnahmefällen – auch in realen Prozessen möglich ist.

Ein Großteil der Fertigungsprozesse fällt in eine dieser beiden Klassen, so dass die entwickelten Methoden auf eine Vielzahl unterschiedlicher Prozesse angewendet werden können. Die Problemklassen werden zur Untersuchung der entwickelten Methoden anhand von Umformprozessen in der Metallverarbeitung instanziiert.

# 2 Stand der Wissenschaft

In diesem Kapitel wird der Stand der Wissenschaft mit Bezug auf die vorliegende Arbeit aufbereitet. Dabei werden verwandte und grundlegende Arbeiten und Methoden vorgestellt und formal eingeführt.

## 2.1 Wissenschaftlicher Rahmen der Arbeit

In direkter Beziehung zu der vorliegenden Arbeit stehen Arbeiten die im Vorfeld von Melanie Senn [1–3] und Susanne Witt (geborene Fischer) [4, 5] im Rahmen des Graduiertenkollegs 1483 durchgeführt wurden. Diese Arbeiten beschäftigen sich, wie die vorliegende Arbeit auch, im Schwerpunkt mit der Anwendung maschineller Lernverfahren auf Fertigungsprozesse. Wie bei der vorliegenden Arbeit werden entwickelte Methoden dabei insbesondere anhand simulierter Tiefziehprozesse untersucht. Ein Großteil der Arbeiten wird anhand eines 2D Modells zur Simulation eines Tiefziehprozesses mittels der Finite-Elemente-Methode (siehe Abschnitt 4.3.1) erprobt. Das Simulationsmodell zeichnet sich durch eine hohe Rechenperformanz aus, und ermöglicht so umfangreiche Untersuchungen datengetriebener Methoden. Im Folgenden wird dieses Modell kurz als *2D Tiefziehmodell* bezeichnet.
Im Mittelpunkt der Arbeiten von Melanie Senn steht der Einsatz von maschinellem Lernen zur Prozessbeobachtung und optimalen Regelung von Fertigungsprozessen. In [1] werden Regressionsmethoden und Methoden zur Dimensionsreduktion zur Prozessbeobachtung und zur Vorhersage bestimmter

Eigenschaften der Prozessergebnisse untersucht. Trainiert werden die Methoden anhand eines 100 Stichproben umfassenden Datensatzes einer Multiskalensimulation. Die Simulation integriert ein Mikrostrukturmodell mittels Homogenisierung in das 3D Simulationsmodell eines Tiefziehprozesses [6]. Die Stichproben werden mithilfe des Modells unter Variation des Reibungskoeffizienten und der Niederhaltekraft erstellt. Die Vorhersagegenauigkeit des Prozessbeobachters wird durch den Vergleich mit Realexperimentergebnissen evaluiert. Hierbei wird die Ausprägung der *Zipfelbildung* mittels des Prozessbeobachters vorhergesagt und Vorhersageergebnisse mit Ergebnissen der Multiskalensimulation und Ergebnissen von Realexperimenten verglichen. In [2] werden verschiedene Ansätze des *approximate dynamic Programming* vorgestellt und anhand der optimalen Regelung eines simulierten Tiefziehprozesses untersucht. Die Optimierung findet *offline* auf Basis einer Stichprobenmenge von Prozessdaten statt. Ähnlich wie in der vorliegenden Arbeit werden im Rahmen des *approximate dynamic Programming* künstliche neuronale Netze zur Approximation der Erwartungswerte der zukünftigen Belohnung gelernt. Darüber hinaus finden in [2] künstliche neuronale Netze auch Verwendung zur Approximation eines deterministischen Zustandsübergangsmodells. Vergleichend evaluiert werden die Methoden anhand des 2D Tiefzieh-Modells mit verrauschten Zustandsübergängen. Eine in [2] eingeführte Methode, *Backward Approximate Dynamic Programming* wird in 2.2.3 ausführlich besprochen.
Durch Susanne Witt wurden die Arbeiten zur Prozessbeobachtung weiter vertieft. In [4] wird eine Methode zur nicht-linearen Dimensionsreduktion vorgestellt, bei der die Merkmale im dimensionsreduzierten Raum geordnet nach Wichtigkeit vorliegen. Die Methode verwendet eine Menge sequentiell gelernter *Autoencoder* (siehe [7], Kapitel 14) mit jeweils einem Neuron in der mittleren, sogenannten *Flaschenhals-Schicht*. Die Methode wird anhand einer Stichprobenmenge des 2D Tiefzieh-Modells evaluiert. Die Stichprobenmenge wird unter Variation der Niederhaltekraft erzeugt. Der Prozesszustand ist in [4] durch die *Von-Mises-Vergleichsspannungen* an den Integrationspunkten des Simulationsmodells repräsentiert. Darauf aufbauend wurde ein Prozessmodell

mittels *symbolischer Regression* gelernt [5]. Das Ziel bei der Verwendung von *symbolischer Regression* ist hier die Interpretierbarkeit des gelernten Prozessmodells. Auch hier dient die Gesamtheit der *Von-Mises-Vergleichsspannungen* an den Integrationspunkten des 2D Tiefzieh-Modells als Prozesszustand.
Ein zweiter, in Kapitel 5 vorgestellter, Teil der Arbeit wurde im Rahmen des DFG Projektes „Maßgeschneiderte Werkstoffeigenschaften durch Mikrostrukturoptimierung“ durchgeführt. Das hierbei verwendete Taylor-Materialmodell und die darauf beruhende Simulation des uniaxialen Deformationsprozesses wurde von Lukas Morand am *Fraunhofer Institut für Werkstoffmechanik IWM* entwickelt [8] und basiert auf einem Kristallplastizitätsmodell, das von Jan Pagenkopf, ebenfalls am Fraunhofer IWM, entwickelt wurde [9].

## 2.2 Optimierung von Entscheidungsprozessen

Zu Beginn dieses Abschnitts werden in 2.2.1 Markov-Entscheidungsprozesse als formale Grundlage der Problemdefinition bei der Entscheidungsoptimierung eingeführt. Anschließend werden in 2.2.2 bis 2.2.7 allgemeine bewertungsbasierte Lösungsmethoden der dynamischen Programmierung und des bestärkenden Lernens vorgestellt, welche von besonderer Bedeutung für die vorliegende Arbeit sind. In 2.2.8 wird der Fokus geweitet, indem die zuvor im Detail vorgestellten Algorithmen eingeordnet und alternative Lösungsansätze skizziert werden.

### 2.2.1 Markov-Entscheidungsprozesse

Markov-Entscheidungsprozesse[1] (Auch Markov-Entscheidungsprobleme, engl. *Markov Decision Processes*, MDP) bilden die formale Grundlage zur Beschreibung zeit-diskreter Entscheidungsprozesse unter stochastischen Bedingungen. Grundlegend bestehen Markov-Entscheidungsprozesse aus einer Menge von Aktionen $a \in A$, einer Menge von Zuständen $s \in S$ und einer Zustandsübergangsfunktion $P : S \times A \times S \to [0,1]$, wobei $P(s_t, a_t, s_{t+1}) = Pr(s_{t+1}|s_t, a_t)$ die Wahrscheinlichkeit angibt, mit der durch die Aktion $a_t \in A$ zum Zeitschritt $t \in \mathbb{N}_0^+$ ein Übergang von Zustand $s_t \in S$ in den Nachfolgezustand $s_{t+1} \in S$ ausgelöst wird. Zustände $s \in S$ beinhalten jegliche für den weiteren Prozessverlauf relevante Information über die Prozessvergangenheit. Markov-Entscheidungsprozesse sind damit gedächtnislos und die Prozessdynamik ist durch $P$ vollständig charakterisiert. Diese Eigenschaft wird als *Markov-Eigenschaft* bezeichnet. Zustände $s$ eines Markov-Entscheidungsprozesses werden in Anlehnung daran auch als *Markov-Zustände* bezeichnet. Der Aktionsraum $A$ wird in dieser Arbeit als statische Menge betrachtet, grundsätzlich erlaubt die Formulierung des Markov-Entscheidungsprozesses auch die Definition der Menge verfügbarer Aktionen $A_s$ in Abhängigkeit des aktuellen Zustands $s$.
Eine Belohnungsfunktion $R : S \times A \times S \to \mathbb{R}$ bewertet den Zustandsübergang von Zustand $s_t$ in Zustand $s_{t+1}$ via $a_t$ durch eine skalare Größe $R(s_t, a_t, s_{t+1})$. In der Praxis, wie auch an vielen Stellen dieser Arbeit, ist die Belohnung häufig indifferent bezüglich des Ausgangszustands $s_t$ und der Aktion $a_t$, so dass die Belohnungsfunktion lediglich über die erreichten Zustände $s_{t+1}$ definiert ist: $R : S \to \mathbb{R}$. Im Folgenden wird $R$ als beschränkte Abbildung angenommen.

[1] Benannt nach Andrei Andrejewitsch Markow, im englischen Andrey Andreyevich Markov. In dieser Arbeit verwendete Schreibweise ist Markov, in Übereinstimmung mit dem überwiegenden Teil der Fachliteratur.

Das Ziel bei der Lösung eines Markov-Entscheidungsprozesses ist die Maximierung der erwarteten Summe der zukünftigen diskontierten Belohnungssignale durch das Treffen von Aktions-Entscheidungen $a$. Für einen exemplarischen, endlichen, Pfad $s_t, a_t, s_{t+1}, a_{t+1}, ..., s_T$ aus Zuständen und Aktionen, beträgt die Summe $\hat{r}$ der zukünftigen diskontierten Belohnungssignale, im Folgenden kurz *Ertrag* genannt, zum Zeitpunkt $t$

$$\hat{r} = \sum_{k=t}^{T-1} \gamma^k R(s_k, a_k, s_{k+1}). \tag{2.1}$$

Ein Diskontierungsfaktor (engl. *discount Factor*) $\gamma \in [0,1]$ bestimmt dabei die Gewichtung der Belohnungen in Abhängigkeit vom Zeitpunkt des Auftretens. Für den Fall $T \to \infty$ und $\gamma = 1$ konvergiert $\hat{r}$ im allgemeinen Fall nicht. Bei sogenannten *Entscheidungsprozessen mit unendlichem Zeithorizont* ist deshalb eine alternative Problemformulierung von Bedeutung, bei der anstelle des erwarteten Ertrags die mittlere zukünftige Belohnung maximiert wird (vgl. *R-Learning* [10]). Diese Formulierung ermöglicht die Erwartungswertbildung bei fortlaufenden Belohnungen für $\gamma = 1$. Die in dieser Arbeit betrachteten Problemstellungen haben die Gemeinsamkeit, dass es sich um sogenannte *Entscheidungsprozesse mit endlichem Zeithorizont* handelt, bei denen nach einer endlichen Anzahl von Zeitschritten garantiert ein Endzustand erreicht ist. *Entscheidungsprozesse mit unendlichem Zeithorizont* spielen in späteren Teilen der Arbeit keine weitere Rolle und werden aus diesem Grund im Folgenden nicht weiter behandelt.

*Markov-Entscheidungsprozesse mit endlichem Zeithorizont* sind formal definiert durch das Tupel $(S, A, P, R, \gamma, P_0, \bar{S})$, wobei $P_0 : S \to [0,1]$ die Verteilung der Anfangszustände und $\bar{S} \subseteq S$ die Menge der Endzustände angibt. *Entscheidungsprozesse mit festem Zeithorizont* sind ein Spezialfall der *Entscheidungsprozesse mit endlichem Zeithorizont*, mit $\bar{S} = S_T$, wobei $S_t \subseteq S$ die Menge der in Zeitschritt $t$ erreichbaren Zustände definiert. *Entscheidungsprozesse mit festem Zeithorizont* sind durch das Tupel $(S, A, P, R, \gamma, P_0, T)$ formal definiert. An

einigen Stellen werden zur übersichtlicheren Darstellung Zustände $s_t$ ohne Verwendung des Zeit-Indexes durch $s$ und Nachfolgezustände $s_{t+1}$ durch $s'$ symbolisiert. Auf gleiche Art repräsentiert $a'$ die Nachfolgeaktion $a_{t+1}$. Bei der Anwendung ist ein Zustand $s$ üblicherweise durch einen Vektor $\mathbf{s} \in \mathbb{R}^n$ in einem reellen Zustandsraum $S \subseteq \mathbb{R}^n$ beschrieben, während Aktionen $a \in A$ im Rahmen dieser Arbeit Elemente einer endlichen Aktionsmenge $A$ darstellen. Anstelle der Zustände sind in Teilen der Arbeit Vektoren beobachtbarer Größen $\mathbf{o} \in \mathbb{R}^n$ gegeben, die den Prozesszustand nicht im Sinne der *Markov-Eigenschaft* charakterisieren. Dieser Fall der sogenannten *partiellen Beobachtbarkeit* wird in 2.3.1 behandelt.

Verfahren zur Lösung von Markov-Entscheidungsprozessen entstammen zwei ursprünglich weitgehend getrennten wissenschaftlichen Strömungen: der optimalen Regelung (dynamische Programmierung) und dem maschinellen Lernen (bestärkendes Lernen). Diese Trennung findet sich an einigen Stellen der verwandten Arbeiten in der Notation wieder. Während im Bereich des bestärkenden Lernens, wie auch in dieser Arbeit, Zustände durch das Symbol $s$ (für engl. 'state') Zustandsräume durch das Symbol $S$ (für engl. 'state space'), Aktionen durch $a$ (für engl. 'action') und Aktionsräume durch das Symbol $A$ (für engl. 'action space') repräsentiert werden, orientiert sich die Notation der dynamischen Programmierung an der gängigen Notation der Regelungstechnik. Zustände sind hier durch $x$ (Symbol für den Zustandsvektor in der Regelungstechnik ), sowie Aktionen durch $u$ (Symbol für die Stellgröße in der Regelungstechnik ) repräsentiert. Das Ziel ist im bestärkenden Lernen als die Maximierung der zukünftig erwarteten Belohnung $R$ formuliert. Die übliche Formulierung des Ziels in Arbeiten der dynamischen Programmierung ist die Minimierung zukünftig erwarteter Kosten $J$.

### 2.2.2 Dynamische Programmierung

Zur Maximierung des erwarteten Ertrags $\hat{r}$ wird eine Entscheidungs-Strategie (engl. *Policy*) $\pi : A \times S \to [0,1]$ optimiert. Die Strategie weißt einem Zustand des Entscheidungsprozesses $s \in S$ für jede wählbare Aktion $a \in A$ die Wahrscheinlichkeit zu mit der diese gewählt wird $\pi(a,s) = Pr(a|s)$. Für eine Strategie $\pi$ gibt die Zustands-Bewertungsfunktion $V_\pi : S \to \mathbb{R}$ (engl. *State-Value Function*) für Zustände $s \in S$ den Erwartungswert des zukünftigen diskontierten Ertrags an. $V_\pi$ ist formal definiert als

$$V_\pi(s_t) = \mathbb{E}_{\pi,P}\left[\sum_{k=t}^{T-1} \gamma^k R(s_k, a_k, s_{k+1})\right]. \tag{2.2}$$

Die Verteilung der Aktionen $a_k$ folgt dabei der Strategie $\pi(a_k, s_k)$. Die Verteilung der Folgezustände $s_{t+1}, ..., s_T$ folgt der Zustandsübergangsfunktion. Der Erwartungswert eines Endzustands $\bar{s} \in \bar{S}$ ist definiert als $V_\pi(\bar{s}) = 0$. Die Zustands-Bewertungsfunktion $V_\pi(s_t)$ kann in rekursiver Form ausformuliert werden:

$$\begin{aligned} V_\pi(s_t) &= \sum_{a_t \in A} \pi(s_t, a_t) \sum_{s_{t+1} \in S} P(s_t, a_t, s_{t+1}) \big[R(s_t, a_t, s_{t+1}) + \gamma V_\pi(s_{t+1})\big] \\ &= \mathbb{E}_{\pi,P}\big[R(s_t, a_t, s_{t+1}) + \gamma V_\pi(s_{t+1})\big]. \end{aligned} \tag{2.3}$$

Diese, *Bellman Gleichung* genannte, rekursive Formulierung stellt die Basis für die Formulierung der *Bellman-Optimalitätsgleichungen* und damit der Lösung von Markov-Entscheidungsprozessen durch dynamische Programmierung und Methoden des bestärkenden Lernens dar. Eine Lösung des Markov-Entscheidungsprozesses ist eine optimale Strategie $\pi^*$. Diese ist gefunden, wenn für alle $s \in S$ gilt

$$\pi^* = \arg\max_\pi V_\pi(s). \tag{2.4}$$

Die zugehörige optimale Zustands-Bewertungsfunktion $V^*(s)$ entspricht

$$V^*(s) = \max_{\pi} V_\pi(s). \tag{2.5}$$

Die *Bellman-Optimalitätsgleichung* für $V^*$ folgt hieraus in Kombination mit der rekursiven Formulierung der Zustands-Bewertungsfunktion in (2.3) und ist gegeben durch

$$V^*(s_t) = \max_{a_t \in A} \sum_{s_{t+1} \in S} P(s_t, a_t, s_{t+1}) \left[ R(s_t, a_t, s_{t+1}) + \gamma V^*(s_{t+1}) \right]. \tag{2.6}$$

Im Folgenden wird zwischen stochastischen Strategien und deterministischen Strategien $\bar{\pi} : S \to A$ unterschieden. Bei der Formulierung des Optimierungsziels als Maximierung des erwarteten Ertrags und unter den oben genannten weiteren Annahmen[2] ist bekannt, dass wenn eine optimale Strategie des Entscheidungsprozesses existiert, auch mindestens eine deterministische Strategie $\bar{\pi}^* : S \to A$ existiert (siehe [11], Kapitel 6). Wenn $V^*$ und $P$ bekannt ist, lässt sich $\bar{\pi}^*$ auf einfache Art bestimmen:

$$\bar{\pi}^*(s_t) = \arg\max_{a_t \in A} \sum_{s_{t+1} \in S} P(s_t, a_t, s_{t+1}) \left[ R(s_t, a_t, s_{t+1}) + \gamma V^*(s_{t+1}) \right] \tag{2.7}$$

Für $s \in S$ lässt sich aus (2.6) ein Gleichungssystem aus $|S|$ Gleichungen mit $|S|$ unbekannten aufstellen. Dieses ist theoretisch durch Lösungsverfahren für nichtlineare Gleichungssysteme lösbar, wenn der Entscheidungsprozess, insbesondere die Zustandsübergangsfunktion $P$, bekannt ist. In der Praxis ist dies aufgrund von Beschränkungen der Rechenzeit und des verfügbaren Speichers unmöglich und Methoden der dynamischen Programmierung werden zur Approximation der *Bellman-Optimalitätsgleichung* verwendet.

[2] (a) Die Belohnungsfunktion $R$ ist eine beschränkte Funktion. (b) Die Menge der ausführbaren Aktionen $A$ ist abzählbar endlich.

POLICYITERATION$(P, R, \gamma, S)$

1 Initialize $\bar{\pi}$ and $V$
2 **repeat**
3 **for** $s_t \in S$:
4 $V(s_t) \leftarrow \sum_{s_{t+1} \in S} P(s_t, \bar{\pi}(s_t), s_{t+1}) \left[ R(s_t, \bar{\pi}(s_t), s_{t+1}) + \gamma V(s_{t+1}) \right]$
5 **until** $V$ converges (to $V_{\bar{\pi}}$)
6 **for** $s_t \in S$:
7 $\bar{\pi}' \leftarrow \arg\max_{a_t \in A} \sum_{s_{t+1} \in S} P(s_t, a_t, s_{t+1}) \left[ R(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}) \right]$
8 **if** $\bar{\pi}' = \bar{\pi}$:
9 $\bar{\pi}^* = \bar{\pi}$
10 **else** :
11 $\bar{\pi} \leftarrow \bar{\pi}'$
12 go to Line 2

Listing 1. Policy Iteration [12]

### 2.2.3 Grundlegende Verfahren der dynamischen Programmierung

Grundlegende Algorithmen der dynamischen Programmierung zur Lösung von Markov-Entscheidungsprozessen sind der *Policy Iteration* Algorithmus [12] (Listing 1) und der *Value Iteration* Algorithmus [13] (Listing 2). Beide verfeinern iterativ eine Approximation $V \approx V^*$ der optimalen Zustands-Bewertungsfunktion $V^*$. $V$ konvergiert bei beiden Ansätzen für die hier betrachteten *Entscheidungsprozesse mit endlichem Zeithorizont* garantiert zu $V^*$. Eine optimale Strategie $\bar{\pi}^*$ kann (2.7) folgend aus $V^*$ extrahiert werden. Die initiale Bewertungsfunktion $V$, und im Fall von *Policy Iteration* auch die initiale Strategie $\bar{\pi}$, kann dabei beliebig gewählt werden, wobei beachtet werden muss, dass per Definition $V(\bar{s}) = 0$ für alle Endzustände $\bar{s} \in \bar{S}$ gilt.

*Policy Iteration* (Listing 1) besteht aus der iterativen Ausführung von zwei aufeinanderfolgenden Schritten: (a) Der Berechnung der Bewertungsfunktion $V_{\bar{\pi}}$ der aktuellen Strategie $\bar{\pi}$ (in blauer Farbe dargestellt, Zeilen 2 bis 5) und (b)

VALUEITERATION$(P,R,\gamma,S)$
1 Initialize $V$
2 **repeat**
3 **for** $s_t \in S$:
4 $V(s_t) \leftarrow \max_{a_t \in A} \sum_{s_{t+1} \in S} P(s_t,a_t,s_{t+1}) \left[R(s_t,a_t,s_{t+1}) + \gamma V(s_{t+1})\right]$
5 **until** $V$ converges (to $V^*$)
6 Extract $\bar{\pi}^*$ following (2.7)

Listing 2. Value Iteration [13]

der Extraktion einer verbesserten Strategie $\bar{\pi}'$ (in roter Farbe dargestellt, Zeilen 6 und 7) aus $V_{\bar{\pi}}$. Dem *Policy Improvement Theorem* [14] folgend gilt für das Ergebnis der Schritte (a) und (b) in jedem Fall $V_{\bar{\pi}'(s)} \geq V_{\bar{\pi}(s)}$ für alle Zustände $s$ in $S$. Die optimale Strategie $\bar{\pi}^*$ ist garantiert erreicht, wenn in Schritt (b) keine verbesserte Strategie gefunden werden kann. Anderenfalls wird $\bar{\pi}$ durch $\bar{\pi}'$ ersetzt und eine weitere Iteration wird initiiert (Zeilen 10 bis 12).
Der *Value Iteration* Ansatz (Listing 2) kombiniert die Schritte (a) und (b) in einem kombinierten Update-Schritt (Zeile 4). Hierbei wird die Bewertungsfunktion $V_{\bar{\pi}}$ nicht mehr vollständig berechnet bevor $\bar{\pi}$ extrahiert wird. Stattdessen wird $\bar{\pi}$ an der Stelle $\bar{\pi}(s_t)$ implizit aktualisiert, indem für $s_t$ die Aktion $a_t \in A$ gesucht wird für die der Erwartungswert der Belohnung maximal ist. Der Erwartungswert $V(s_t)$ wird an der Stelle $s_t$ für die derart implizit aktualisierte Strategie aktualisiert. Dieses lokale Update wird für alle Zustände $s_t$ (Zeile 3) wiederholt (Zeilen 2, 5) durchgeführt. Der beschriebene Zusammenhang wird klar, wenn man das *Value Iteration* Update aus Zeile 4 zerlegt:

$$\bar{\pi}' \leftarrow \arg\max_{a_t \in A} \sum_{s_{t+1} \in S} P(s_t,a_t,s_{t+1}) \left[R(s_t,a_t,s_{t+1}) + \gamma V(s_{t+1})\right] \tag{2.8}$$

entspricht dann der lokalen Anwendung von Schritt (b) und

$$V(s_t) \leftarrow \sum_{s_{t+1} \in S} P(s_t,\bar{\pi}'(s_t),s_{t+1}) \left[R(s_t,\bar{\pi}'(s_t),s_{t+1}) + \gamma V(s_{t+1})\right] \tag{2.9}$$

der lokalen Anwendung von Schritt (a) des *Policy Iteration* Algorithmus. Die Grundidee des kombinierten lokalen Updates findet sich in gleicher oder ähnlicher Form bei allen im Folgenden beschriebenen und in der Arbeit entwickelten Algorithmen wieder.

Zur Konvergenz der bisher vorgestellten Algorithmen muss das jeweilige Update der Zustands-Bewertungsfunktion $V$ pro Zustand $s \in S$ in der Regel mehrfach angewandt werden, da das jeweilige Update wiederum auf approximierten Bewertungen $V(s_{t+1})$ der Zustände des nächsten Zeitschritts beruht. Im Fall von *Entscheidungsprozessen mit endlichem Zeithorizont* können Zustände $s \in S$ in eine endliche Menge zeitlich geordneter Zustandsmengen $S_0, ..., S_T$ gebracht werden, wobei $S_t \subseteq S$ die Menge der in Zeitschritt $t$ erreichbaren Zustände ist. Dies ermöglicht, für Zeitschritt-abhängige Zustands-Bewertungsfunktionen $V_t$ das Update ausgehend von den Endzuständen $S_T$ rückwärts in den Zeitschritten anzuwenden und so sicherzustellen, dass zum Zeitpunkt des Updates von $V_t(s_t)$ bereits $V_{t+1}(s_{t+1}) = V^*_{t+1}(s_{t+1})$ gilt. *Backward Dynamic Programming* (vgl. [15] S. 65) ist eine Spezialform des *Value Iteration* Algorithmus und folgt diesem Ansatz. Die Anzahl der zur Konvergenz benötigten Updates wird hierdurch auf $\sum_t |S_t|$ beschränkt. Wie in Listing 3 dargestellt, genügt hierbei ein einmaliger Durchlauf der Zeitschritt-abhängigen Zustandsräume $S_T, ..., S_0$ in umgekehrter zeitlicher Reihenfolge.

Die besprochenen Algorithmen lösen Markov-Entscheidungsprozesse in polynomieller Zeit [16] (in Abhängigkeit der Kardinalitäten von $A$ und $S$). Üblicherweise repräsentieren Zustände einen Punkt in einem Zustandsraum. Die Anzahl der zur Repräsentation des Raumes benötigten Zustände $|S|$ steigt exponentiell mit der Dimension des Zustandsraums (beziehungsweise der Anzahl der repräsentierten Zustandsvariablen). Dieser *Curse of Dimensionality* [14] führt dazu, dass die hier beschriebenen klassischen Verfahren in vielen Realanwendungen aufgrund begrenzter Rechen- und Speicher-Ressourcen nicht anwendbar sind. Darüber hinaus ist die Zustandsbeschreibung mittels kontinuierlicher Zustandsvariablen nicht mit einer Begrenzung der Zustandsmenge $S$ vereinbar. Ein Ansatz zum Umgang mit diesen Problemen ist die Verwendung

BACKWARDDP$(P, R, \gamma, [S_0, ..., S_T], \bar{S})$

1 **for** $t \in T, ..., 0$:
2 **for** $s_t \in S_t$:
3 **if** $s_t \in \bar{S}$:
4 $V_t^*(s_t) \leftarrow 0$
5 **else**
6 $V_t^*(s_t) \leftarrow \max_{a_t \in A} \sum_{s_{t+1} \in S} P(s_t, a_t, s_{t+1}) \left[ R(s_t, a_t, s_{t+1}) + \gamma V_{t+1}^*(s_{t+1}) \right]$
7 Extract $\bar{\pi}^*$ following (2.7)

Listing 3. Backward Dynamic Programming (nach [15])

von Methoden des überwachten Maschinellen Lernens zur Approximation der Bewertungsfunktionen.

### 2.2.4 Approximative Verfahren der dynamischen Programmierung

Die im vorangegangenen Unterabschnitt beschriebenen klassischen Verfahren haben die Gemeinsamkeit, dass die Zustands-Bewertungsfunktion $V$ explizit in tabellarischer Form $(S \times A)$ repräsentiert ist. Dies führt, wie besprochen, zu Problemen, wenn die Kardinalität der Zustände $|S|$ hoch ist, oder gar von kontinuierlichen Zustandsbeschreibungen $\mathbf{s} \in \mathbb{R}^n$ auszugehen ist. Eine Möglichkeit mit diesen Problemen umzugehen ist die Verwendung von Regressionsverfahren zur Approximation der Zustands-Bewertungsfunktion. Im vorangegangenen Unterabschnitt angeführte Garantien bezüglich der Konvergenz der klassischen Algorithmen sind bei der Verwendung von Funktionsapproximationsmethoden nicht länger gültig.

BACKWARDADP$(P, R, \gamma, [\mathcal{S}_0, ..., \mathcal{S}_T]))$

1 $\mathcal{V}_T(\mathbf{s}_T, \theta_T) \coloneqq 0, \ \forall \mathbf{s}_T \in S_T, \forall \theta_T$
2 **for** $t \in T-1, ..., 0$:
3 $\quad D_t \leftarrow \emptyset$
4 $\quad$ **for** $\mathbf{s}_t \in \mathcal{S}_t$:
5 $\quad\quad x_\mathcal{V} \leftarrow \mathbf{s}_t$
6 $\quad\quad y_\mathcal{V} \leftarrow \max_{a_t \in A} \int \Big[ P(\mathbf{s}_t, a_t, \mathbf{s}_{t+1}) \big( R(\mathbf{s}_t, a_t, \mathbf{s}_{t+1}) + \gamma \mathcal{V}_{t+1}(\mathbf{s}_{t+1}, \theta_{t+1}) \big) \Big] d\mathbf{s}_{t+1}$
7 $\quad\quad D_t \leftarrow D_t \cup \{(x_\mathcal{V}, y_\mathcal{V})\}$
8 $\quad$ Learn Parameters $\theta_t \leftarrow$ based on $D_t$
9 Extract $\bar{\pi} \approx \bar{\pi}^*$ following (2.7) with $\mathcal{V}_t(\mathbf{s}_t, \theta_t) \approx V^*(s_t)$

Listing 4. Backward Approximate Dynamic Programming (nach [2])

Im Folgenden wird stellvertretend die approximative Form des zuvor besprochenen *Backward Dynamic Programming* eingeführt, die wie später besprochen in einigen Punkten Gemeinsamkeiten mit dem in dieser Arbeit entwickelten *Backward Fixed Horizon Neural Q-Learning* Algorithmus aufweist (siehe Abschnitt 4.2).

*Backward Approximate Dynamic Programming* (Listing 4) wurde von Melanie Senn in [2] vorgestellt und ist ein Verfahren zur approximativen Lösung von *Entscheidungsprozessen mit festem Zeithorizont* und potenziell kontinuierlichem Zustandsraum. Ein Modell der Zustandsübergangsfunktion $P$ und der Belohnungsfunktion $R$ wird dabei als gegeben angenommen. Außerdem liegt eine repräsentative Menge von Zustandsbeschreibungen $\mathbf{s}_t \in \mathcal{S}_t$ pro Zeitschritt $t$ vor. An die Stelle der Zeitschritt-abhängigen tabellarischen Zustands-Bewertungsfunktionen $V_t$ tritt die Approximation $\mathcal{V}_t(\mathbf{s}_t, \theta_t) \approx V_t(s_t)$ mit eigenständigen Modellparametern $\theta_t$ pro Zeitschritt $t \in 0, ..., T-1$. Für Zustands-Bewertungen der Endzustände gilt auch hier $\mathcal{V}_T(\mathbf{s}_T, \theta_T) = 0$ für alle $\mathbf{s}_T$ und $\theta_T$ (Zeile 1) . Ausgehend von $t = T$ wird rückwärts in den Zeitschritten (Zeile 2) ein Datensatz $D_t$ zum Training der Modellparameter $\theta_t$ erstellt (Zeilen 3 bis

7) und anschließend die Funktionsapproximation $\mathcal{V}(\mathbf{s}_t, \theta_t)$ trainiert (Zeile 8). Eingangsdaten der Funktionsapproximation $x_\mathcal{V} \in \mathbb{R}^n$ bilden die Zustandsbeschreibungen $\mathbf{s}_t$, Zielgrößen $y_\mathcal{V} \in \mathbb{R}$ sind die in Zeile 6 berechneten Zustands-Bewertungen. Da die approximative Methode im Fall von kontinuierlichen Zustandsräumen anwendbar sein soll, ist die Bildung einer Summe über die Folgezustände während des Updates (vgl. *Backward Dynamic Programming* Listing 3, Zeile 6) nicht länger möglich. Stattdessen wird unter Anwendung der *Simpsonschen Formel* ein Integral über die Erwartungswerte der Folgezustände approximiert. Neben der Zustands-Bewertungsfunktion werden künstliche Neuronale Netze in [2] auch zur Punktschätzung der Zustandsübergangsfunktion $P$ auf Basis von Prozessdaten genutzt.

### 2.2.5 Bestärkendes Lernen

Bestärkendes Lernen (engl. *reinforcement learning*) ist ein Teilbereich des maschinellen Lernens und hat, wie die oben beschriebene dynamische Programmierung, zum Ziel Markov-Entscheidungsprozesse zu lösen. Im Gegensatz zu den beschriebenen Methoden der dynamischen Programmierung, bei denen die Lösung des Entscheidungsprozesses auf einer bekannten oder approximierten Zustandsübergangsfunktion $P$, der bekannten Belohnungsfunktion $R$ und der Kenntnis des Zustandsraums $S$ beruht, lernen Algorithmen des bestärkenden Lernens die Lösung *online* durch Interaktion mit dem Prozess.
Der Algorithmus des bestärkenden Lernens trifft während des Lernens in aller Regel autonome Entscheidungen zur Interaktion mit dem Prozess und führt diese durch ansprechen einer geeigneten Schnittstelle aus. Ein System, bestehend aus dem Algorithmus des bestärkenden Lernens und in einigen Fällen virtuellen oder physischen Komponenten zur Wahrnehmung und zur Interaktion mit der Umgebung wird daher gängiger Weise als *Agent* [17] bezeichnet. Die sogenannte *Umgebung* besteht aus einem potenziell stochastischen

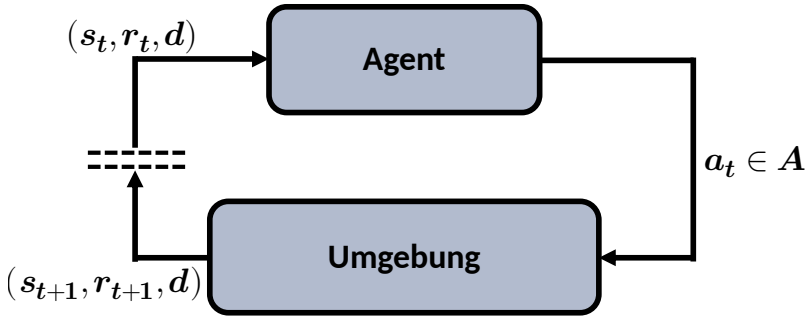



Abbildung 2.1: Interaktions-Zyklus des bestärkenden Lernens (nach Sutton, Barto [16])

Prozess, einem Belohnungssignal-Geber und einem Beobachter. Prozess, Belohnungssignal-Geber und Beobachter bilden den Markov-Entscheidungsprozess ab. Die Interaktion zwischen Agent und Umgebung ist in Abbildung 2.1 skizziert. Der Prozess befindet sich zum Zeitpunkt $t$ in Zustand $s_t$ und nimmt einer unbekannten Zustandsübergangsfunktion $P$ folgend, durch äußere Anregung $a_t \in A$ einen Nachfolgezustand $s_{t+1} \in S$ an. Der Belohnungssignal-Geber emittiert zum Zeitpunkt $t+1$ einer unbekannten Belohnungsfunktion $R$ folgend ein numerisches Belohnungssignal $r$ für den Zustandsübergang von $s_t$ zu $s_{t+1}$ durch die Aktion $a$. Der Beobachter gibt gleichzeitig den Markov-Zustand $s_t$ oder die Markov-Zustandsbeschreibung $\mathbf{s}_{t+1}$ zurück.

Bei *Entscheidungsprozessen mit endlichem Zeithorizont* ist der Lernvorgang in *Episoden* genannte Abschnitte eingeteilt. Als Episode $e$ wird eine einzelne Prozessausführung, also die Interaktion ausgehend von dem Startzustand $s_0$ bis zum Auftreten eines Endzustands $\bar{s} \in \bar{S}$ bezeichnet. Da $\bar{S}$ nicht bekannt ist, wird im Fall von *episodischem Lernen* zusätzlich durch die Umgebung der boolesche Wert $d$ zur Signalisierung des Episodenendes ausgegeben.

Im Folgenden wird eine Auswahl von tabellarischen und approximativen Algorithmen des bestärkenden Lernens vorgestellt, die ähnlich den oben vorgestellten Algorithmen der dynamischen Programmierung durch die Annäherung von Bewertungsfunktionen lernen (engl. *Value-based Methods*) und in Beziehung

zu den im Rahmen der Arbeit entwickelten Algorithmen stehen. Aufgrund der Ausrichtung der Arbeit werden Varianten der jeweiligen Algorithmen zur Lösung von *Entscheidungsprozessen mit endlichem Zeithorizont* vorgestellt. Die Generalisierung zur Anwendbarkeit auf *Entscheidungsprozesse mit unendlichem Zeithorizont* ist jeweils durch geringfügige, hier nicht weiter behandelte, Anpassungen möglich (siehe [16]).

### 2.2.6 Tabellarisches bestärkendes Lernen

Bei den oben vorgestellten Algorithmen der dynamischen Programmierung werden Markov-Entscheidungsprozesse durch das iterative verbessern von Zustands-Bewertungsfunktionen $V(s)$ (In Listing 1 bis 4) oder approximierten Zustands-Bewertungsfunktionen $\mathcal{V}(\mathbf{s},\theta)$ gelöst. Die jeweiligen Updates von $V(s)$ bzw. $\mathcal{V}(\mathbf{s},\theta)$ haben gemeinsam, dass sie auf den Bewertungen der potenziellen Nachfolgezustände $V(s')$ bzw. $\mathcal{V}(\mathbf{s}',\theta)$ für alle potenziellen Aktionen $a \in A$ beruhen. Voraussetzung hierfür ist Kenntnis über die Zustandsübergangsfunktion $P(s'|s,a)$. Im Fall der beschriebenen modellfreien Optimierung mittels bestärkenden Lernens ist $P$ nicht bekannt und die Updates der Bewertungsfunktion (Zeilen 4 und 7 Listing 1, Zeile 4 Listing 2, Zeile 6 Listing 3, Zeile 6 Listing 4) können nicht durchgeführt werden. Die sogenannte Q-Funktion $Q : S \times A \to \mathbb{R}$ (auch Aktions-Bewertungsfunktion) bewertet Zustands-Aktions-Paare und berücksichtigt so auch die Dynamik des Systems bei dem Übergang des Zustands $s$ zu $s'$. Die Q-Funktion $Q_\pi$ für die Strategie $\pi$ modelliert den Erwartungswert der zukünftigen diskontierten Belohnung für die Strategie $\pi$ und ist in ihrer rekursiven Form definiert als

$$Q_\pi(s_t,a_t) = \mathbb{E}_{P,\pi}\left[R(s_t,a_t,s_{t+1}) + \gamma Q_\pi(s_{t+1},a_{t+1}))\right], \tag{2.10}$$

wobei $a_{t+1}$ der Strategie $\pi$, und die Verteilung des Folgezustands $s_{t+1}$ der unbekannten Zustandsübergangsfunktion $P(s_t, a_t, s_{t+1})$ folgt. Bei *Entscheidungsprozessen mit endlichem Zeithorizont* gilt analog zur Zustands-Bewertungsfunktion $Q(\bar{s}, a) = 0$ für Endzustände $\bar{s} \in \bar{S}$ und alle Aktionen $a \in A$.
Ähnlich wie im Fall der Zustands-Bewertungsfunktion (vgl. (2.6)) kann die *Bellman-Optimalitätsgleichung* der Q-Funktion in rekursiver Form aufgestellt werden:

$$Q^*(s_t, a_t) = \mathbb{E}_P\left[R(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1} \in A} Q^*(s_{t+1}, a_{t+1})\right]. \tag{2.11}$$

Ist $Q^*$ gegeben, ist die Extraktion der optimalen Strategie $\bar{\pi}^*$ auf einfache Art möglich:

$$\bar{\pi}^*(s_t) = \arg\max_{a_t \in A} Q^*(s_t, a_t). \tag{2.12}$$

Anders als im Fall der Zustands-Bewertungsfunktion (vgl. (2.7)) ist hierzu kein Wissen über die Zustandsübergangsfunktion $P$ erforderlich. Die optimale Zustands-Bewertungsfunktion $V^*$ ist für $Q^*$ gegeben durch die Beziehung

$$V^*(s) = \max_{a \in A} Q^*(s, a). \tag{2.13}$$

*Q-Learning* [18] gilt als der Algorithmus der das moderne bestärkende Lernen begründet hat, indem zentrale Ideen aus Bereichen der *optimalen Regelung* (dynamische Programmierung) und der *künstlichen Intelligenz* (Lernen durch Versuch und Irrtum) zusammengebracht wurden (vgl. [16] S. 21). *Q-Learning* bildet bis heute die Grundlage für neue bewertungsbasierte Ansätze des bestärkenden Lernens, so auch für die in 2.2.4 und 2.2.7 vorgestellten approximativen Methoden und die im Rahmen dieser Arbeit entwickelten Methoden.
Der *Q-Learning* Algorithmus ist in Listing 5 für den Fall des episodischen Lernens dargestellt. Initial wird die $Q$-Funktion unter Berücksichtigung von $Q(\bar{s}, a) = 0$ für alle $\bar{s} \in \bar{S}$ und $a \in A$ beliebig initialisiert (Zeile 1). Pro Episode (Zeile 2) wird der Initialzustand beobachtet (Zeile 2). Während der Episode

Q-LEARNING$(\alpha, \tilde{\pi}, n_e)$

```
1  Initialize Q
2  for e = 1 to n_e:
3      observe initial state s
4      repeat
5          execute a following π̃ and observe s', r, d
6          Q(s,a) ← Q(s,a) + α[r + γ max_{a'∈A} Q(s',a') − Q(s,a)]
7          s ← s'
8      until d = 1
```

Listing 5. *Q-Learning* (episodisch) [18]

wird pro Zeitschritt Aktion $a$ durch den Agenten einer Strategie $\tilde{\pi}$ folgend bestimmt und auf der Umgebung ausgeführt. Daraufhin werden der Nachfolgezustand $s'$, das Belohnungssignal $r = R(s,a,s')$ und eine Binärvariable $d$ beobachtet. Die Binärvariable $d$ nimmt den Wert 1 an, wenn $s'$ ein Endzustand und die aktuelle Episode damit beendet ist. Basierend auf den beobachteten Werten $(s,a,r,s')$ findet in Zeile 6 das Update der Q-Funktion statt. Die Lernrate $\alpha$ bestimmt dabei den Einfluss des Updates auf den lokalen Wert $Q(s,a)$.
Die Q-Funktion konvergiert bei ausreichend hohem $n_e$ garantiert gegen $Q^*$, wenn durch die während des Lernens ausgeführte stochastische Strategie $\tilde{\pi} : A \times S \to [0,1]$ (im Folgenden als *Lernstrategie* bezeichnet) sichergestellt ist, dass für jedes $a \in A$ und $s \in S \setminus \bar{S}$ gilt: $\tilde{\pi}(a,s) > 0$ [19].
Diese Bedingung wäre durch eine Lernstrategie $\tilde{\pi}$ erfüllt, durch die Aktionen gleich verteilt ausgeführt werden $\tilde{\pi}(a,s) = 1 \div |A|$. Effizienter sind allerdings Lernstrategien, die die aktuelle Bewertung $Q(s,a)$ zur Priorisierung der Aktionen nutzen. Der Extremfall, der aber nicht mit der Bedingung vereinbar ist, wäre die Nutzung der sogenannten *greedy*-Strategie $\bar{\pi} = \arg\max_{a \in A} Q(s,a)$ als Lernstrategie $\tilde{\pi}$. Die Lernstrategie $\tilde{\pi}(a,s) = 1 \div |A|$ ist eine maximal explorierende Strategie, während $\bar{\pi} = \arg\max_{a \in A} Q(s,a)$ eine maximal verwertende Strategie darstellt. Das Verhältnis zwischen Exploration und Verwertung (*Exploitation*) stellt einen zentralen Zielkonflikt des bestärkenden Lernens dar. Ein

gängiger Umgang mit diesem Zielkonflikt ist die Verwendung einer $\varepsilon$-*greedy* Lernstrategie

$$\tilde{\pi}(a,s) = \begin{cases} 1-\varepsilon & \text{, if } a = \arg\max_{a_i \in A} Q(s,a_i) \\ \frac{\varepsilon}{|A|-1} & \text{, else.} \end{cases} \tag{2.14}$$

Diese stellt eine Mischform der beiden Extrem-Strategien dar, wobei die Explorationsrate $\varepsilon$ das Verhältnis von Exploration und *Exploitation* bestimmt. In der Regel wird $\varepsilon$ in Abhängigkeit von dem aktuellen Lernfortschritt definiert. Zu Beginn des Lernens ist eine hohe Explorationsrate vorteilhaft um den noch unbekannten Entscheidungsprozess zu explorieren. Im Verlauf des Lernens erhöht sich das Wissen über den Prozess und die Q-Funktion nähert sich $Q^*$ an. Eine abnehmende Explorationsrate sorgt nun für zielgerichteteres Lernen. Erreicht werden kann dies beim behandelten episodischen Lernen von *Entscheidungsprozessen mit endlichem Zeithorizont* beispielsweise durch eine exponentielle Annäherung $\varepsilon = \varepsilon_0 \times \exp(-\lambda_\varepsilon e) + \varepsilon_f$ einer initialen Explorationsrate $\varepsilon_0$ an die finale Explorationsrate $\varepsilon_f$, über die Episoden $e$ hinweg.
Der *Q-Learning* Algorithmus ist ein sogenannter *off-Policy* Algorithmus: Die optimale Strategie $\tilde{\pi}^*$ wird gelernt, indem die Approximation $Q \approx Q^*$ iterativ verfeinert wird. Dazu werden Daten verwendet die aus der Interaktion des Agenten mit der Umgebung mittels einer Lernstrategie $\tilde{\pi}$ stammen. Die Lernstrategie $\tilde{\pi}$ und die Strategie deren Q-Funktion gelernt wird stimmen dabei nicht überein. Dies manifestiert sich bei dem Q-Funktionsupdate (Listing 5, Zeile 6), bei dem davon ausgegangen wird, dass $a'$ *greedy* und nicht $\tilde{\pi}$ folgend, gewählt wird. Im Gegensatz dazu ist SARSA [20][3] ein *on-Policy* Algorithmus: die Q-Funktion wird hier für die Lernstrategie $\tilde{\pi}$ gelernt. SARSA entspricht weitgehend dem *Q-Learning* Algorithmus. Anstelle des *Q-Learning Updates*

[3] in [20] unter dem Namen *Modified Connectionist Q-Learning* vorgestellt, der Name SARSA geht auf [21] zurück.

(Listing 5, Zeile 6) ist das SARSA Q-Funktionsupdate (im Weiteren als SARSA-Update bezeichnet) allerdings definiert als

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right], \tag{2.15}$$

wobei $a'$ der tatsächlich ausgeführten Aktion im Nachfolgezeitschritt entspricht.
Durch das beschriebene *SARSA Update* nähert $Q$ die Bewertungsfunktion der Lernstrategie $\tilde{\pi}$ an. Um also die Konvergenz von $Q$ gegen die optimale Bewertungsfunktion $Q^*$ im Fall des *SARSA Updates* sicherzustellen, muss gewährleistet sein, dass $\tilde{\pi}$ im Limit gegen $\pi^*$ konvergiert. Wie im Fall des *Q-Learning Updates* gilt die Konvergenzbedingung $\tilde{\pi}(a,s) > 0$ für alle $a \in A$ und $s \in S \setminus \bar{S}$. Im Fall der oben beschriebenen $\varepsilon$-*greedy* Lernstrategie sind beide Kriterien theoretisch erfüllt, wenn die exponentielle Annäherung der Explorationsrate $\varepsilon$ dem exponentiellen Zerfall ($\varepsilon_f = 0$) entspricht.

### 2.2.7 Approximatives bestärkendes Lernen und Tiefes bestärkendes Lernen

Wie bei der zuvor beschriebenen dynamischen Programmierung werden auch beim bestärkenden Lernen Methoden des überwachten maschinellen Lernens genutzt um durch Approximation der Bewertungsfunktionen die Algorithmen für Realanwendungen mit hoher Kardinalität $|S|$ oder kontinuierlicher Zustandsbeschreibung anwendbar zu machen. Ziel der vorgestellten approximativen Methoden ist die Approximation der Q-Funktion durch künstliche

Neuronale Netze $\mathcal{Q}(\mathbf{s},a,\theta) \approx Q(s,a)$, im Folgenden auch als *Q-Netz* bezeichnet. Zustände $s$ sind durch den Vektor $\mathbf{s} \in \mathbb{R}^n$ repräsentiert, während Aktionen $a$ durch eine skalare Größe $a \in \mathbb{R}$ beschrieben werden[4].

Im Folgenden werden zwei, auf *Q-Learning* und künstlichen Neuronalen Netzen, basierende Methoden des approximativen bestärkenden Lernens vorgestellt: *Neural Fitted Q Iteration* (NFQ) [22] und *Deep Q-Networks* [23]. NFQ stellt ein Schema zum Update der approximierten Q-Funktion beim bestärkenden Lernen dar, das auch durch den in Abschnitt 4.2 entwickelten Algorithmus BFHNQ verfolgt wird. NFQ ist in seiner grundlegenden Form ein sogenannter *Batch*-Algorithmus[5], der basierend auf Erfahrungs-Tupeln $(\mathbf{s},a,r,\mathbf{s}')$ durch Anwendung des *off-Policy Q-Learning Updates* $\mathcal{Q}$-Netze trainiert. *Batch*-Algorithmen wie NFQ können zum *online*-Lernen durch regelmäßiges neu-Trainieren der $\mathcal{Q}$-Netze wie unten beschrieben verwendet werden. Bei *Deep Q-Networks* [23] werden anstelle des regelmäßigen kostenintensiven neu-Trainierens der $\mathcal{Q}$-Netze diese begleitend zum bestärkenden Lernen verfeinernd trainiert. Dadurch wird der Trainingsaufwand deutlich reduziert, sodass auch tiefe Neuronale Netze mit hoher Modellkapazität zum Lernen genutzt werden können. NFQ wird als Basisalgorithmus der in Kapitel 5 vorgestellten Methoden genutzt.

Das *Neural Fitted Q Iteration* Training ist in Listing 6 dargestellt. Auf iterative Art wird hierbei das *Q-Netz* wiederholt neu-trainiert (Zeilen 2 bis 12). Die Initialisierung des künstlichen Neuronalen Netzes kann hierbei beliebig gewählt werden, wobei $\mathcal{Q}(\bar{\mathbf{s}},a,\theta) = 0$ für alle Endzustände $\bar{\mathbf{s}}$ unabhängig von der Aktion $a$ und den Parametern $\theta$ während des gesamten Vorgangs sichergestellt

[4] In der Praxis wird, entgegen der Darstellung, üblicherweise eine Abbildung $\mathcal{Q}(\mathbf{s},\theta)$ von Zuständen $\mathbf{s}$ auf einen Vektor $\mathbb{R}^{|A|}$ der Aktionsweisen Q-Funktionswerte gelernt. Dies ist in der effizienteren Abfrage der Werte begründet und hat keine Auswirkungen auf die beschriebenen Sachverhalte. Zugunsten einer konsistenten Darstellung mit dem tabellarischen Fall wird in dieser Arbeit, wie auch in dem überwiegenden Teil der Fachliteratur, von einer Abbildung $\mathcal{Q}(\mathbf{s},a,\theta) \approx Q(s,a)$ ausgegangen.

[5] verwandt, aber nicht zu verwechseln mit *batch* Optimierungsverfahren beim Training von künstlichen Neuronalen Netzen (siehe Anhang A.0.2)

NFQ($\mathcal{D}, \gamma, n_{\text{nfq}}$)
1 initialize $\mathcal{Q}$
2 **for** $i = 1$ to $n_{\text{nfq}}$:
3 $D \leftarrow \emptyset$
4 **for** $(\mathbf{s}, a, r, \mathbf{s}')$ in $\mathcal{D}$:
5 $x_{\mathcal{Q}} \leftarrow (\mathbf{s}, a)$
6 $y_{\mathcal{Q}} \leftarrow r + \gamma \max_{a' \in \mathbf{A}} \mathcal{Q}(\mathbf{s}', a', \theta)$
7 $D \leftarrow D \cup \{(x_Q, y_{\mathcal{Q}})\}$
8 $\theta \leftarrow$ train $\mathcal{Q}$ based on dataset $D$

Listing 6. *Neural Fitted Q Iteration* (NFQ, episodisch) [22]

sein muss (Zeile 1). Basierend auf den Erfahrungs-Tupeln $(\mathbf{s}, a, r, \mathbf{s}')$ wird ein Trainingsdatensatz $D$ erzeugt (Zeilen 3 bis 7). Die Zielgröße $y_{\mathcal{Q}}$ für $\mathcal{Q}(\mathbf{s}, a, \theta)$ wird dabei analog zu dem *Q-Learning Update* für $Q(s, a)$ (Listing 5, Zeile 6) für eine neutrale Lernrate $\alpha = 1$, basierend auf der aktuellen Approximation $\mathcal{Q}(\mathbf{s}', a', \theta)$ berechnet (Zeile 6). Die Lernrate $\alpha$ des *Q-Learning*-Updates wird aufgrund der Begrenzung der Schrittweite der Gradienten-Updates beim Training des *Q-Netzes* obsolet. Aufgrund des neu-Trainierens der künstlichen Neuronalen Netze kann NFQ mit *Batch* Optimierungsverfahren kombiniert werden (siehe Anhang A.0.2).

NFQ kann aufgrund des *off-Policy* Updates (vgl. 2.2.6) auf einen Datensatz $\mathcal{D}$ mit beliebig erzeugten Erfahrungs-Tupeln angewendet werden, sofern die Erfahrungen aus der Interaktion mit dem Entscheidungsprozess stammen. Das *online*-Lernen durch Interaktion mit dem Prozess wie es in dieser Arbeit behandelt wird entspricht der inkrementellen NFQ Variante (vgl. [22]). Wie beim tabellarischen *Q-Learning* wird hierbei zur Erzeugung der Erfahrungs-Tupel die Lernstrategie $\tilde{\pi}(a, \mathbf{s})$ basierend auf der aktuellen Approximation $\mathcal{Q}(\mathbf{s}, a, \theta)$ ausgeführt. Die Gesamtheit der erzeugten Erfahrungs-Tupel wird in einem anwachsenden *Replay Memory* $\mathcal{D}$ gespeichert. NFQ wird für das *Replay Memory* $\mathcal{D}$ zum Training eines aktualisierten *Q-Netzes* während des bestärkenden Lernens wiederholt ausgeführt.

DQN$(n_e, n_\theta, n_\mathcal{Q}\gamma)$

```
 1  Initialize Q
 2  D ← ∅
 3  for e = 1 to n_e:
 4      observe initial state s
 5      t ← 0
 6      repeat
 7          execute a following π̃ and observe (s', r, d)
 8          D ← D ∪ {(s, a, r, s', d)}
 9          if t divides n_Q:
10              B ← sample experience mini-batch from D
11              θ ← train Q(s, a, θ) by using B following (2.16)
12          if t divides n_θ:
13              θ⁻ ← θ
14          s ← s'
15          t ← t + 1
16      until d = 1
```

Listing 7. *Deep Q Networks* (DQN, episodisch) [23]

Das iterative neu-Trainieren der Approximation $\mathcal{Q}$ stellt einen Trainingsaufwand dar, der mit komplexen Modellen, wie tiefen Neuronalen Netzen, nicht in vertretbarer Rechenzeit durchführbar ist. Der *Deep Q Networks* Algorithmus basiert im Gegensatz dazu auf der fortlaufenden Verfeinerung des *Q-Netzes* durch *mini-Batch* Updates (siehe Anhang A.0.2) während des bestärkenden Lernens.

Der *Deep Q Networks* Algorithmus ist in Listing 7 dargestellt. Nach der Initialisierung des *Q-Netzes* (Zeile 1) und des *Replay Meomory* (Zeile 2) wird DQN für $n_e$ Episoden ausgeführt. Bei der Initialisierung von $\mathcal{Q}$ muss auch hier sichergestellt sein, dass $\mathcal{Q}(\bar{\mathbf{s}}, a, \theta) = 0$ für alle Endzustände $\bar{\mathbf{s}}$ unabhängig von der Aktion $a$ und den Parametern $\theta$ gilt. Wie beim tabellarischen *Q-Learning* werden Aktionen einer Lernstrategie $\tilde{\pi}$ folgend ausgeführt (Zeile 7). Daraufhin

wird der Nachfolgezustand $\mathbf{s}'$, das Belohnungssignal $r$ und die boolesche Variable $d$ zur Identifizierung des Episodenendes zurückgegeben. Wie bei NFQ werden Erfahrungs-Tupel $(\mathbf{s}, a, r, \mathbf{s}', d)$ in einem *Replay Memory* $\mathcal{D}$ gespeichert (Zeile 8). Ein *mini-Batch* Update des $\mathcal{Q}$-Netzes findet alle $n_{\mathcal{Q}}$ Zeitschritte statt (Zeilen 9 bis 11). Zur Bildung des *mini-Batches* werden Erfahrungs-Tupel aus $\mathcal{D}$ ausgewählt (Zeile 10). Zeitlich aufeinanderfolgende Erfahrungs-Tupel sind, auf Ebene der Zustandsvektoren $\mathbf{s}$ und durch die von $\mathcal{Q}$ abgeleitete Strategie, stark korreliert. Das Verwenden der jeweils neuesten Erfahrungen zum Training des $\mathcal{Q}$-Netzes führen zu dem sogenannten *catastrophic Forgetting* Effekt: Erwartungswerte die bereits in zuvor explorierten Bereichen des Zustandsraums gelernt wurden werden durch die neuen Erfahrungen „überschrieben“. Aus diesem Grund werden in einem *Experience Replay* genannten Prozess Erfahrungen zur Erzeugung der Trainingsdaten zufällig gleich-verteilt aus $\mathcal{D}$ gezogen. Für das Erfahrungs-Tupel $(\mathbf{s}, a, r, \mathbf{s}', d)$ ist die Zielgröße für $\mathcal{Q}(\mathbf{s}, a, \theta)$ definiert als

$$y_{\mathcal{Q}} \leftarrow r + \gamma \max_{a' \in \mathbf{A}} \mathcal{Q}(\mathbf{s}', a', \theta^{-}). \tag{2.16}$$

Dies unterscheidet sich von der NFQ Zielgröße (Listing 7, Zeile 6) durch die Entkopplung der Parameter zur Zielgrößenberechnung $\theta^{-}$ von den aktuellen $\mathcal{Q}$-Netz Parametern $\theta$. Da die Q-Funktion $\mathcal{Q}(\mathbf{s}, a, \theta)$ sich, durch das voranschreitende bestärkende Lernen stetig verändert würde eine Kopplung der Parameter dazu führen, dass auch die Zielgrößen einer stetigen Veränderung unterlegen wären. Dies führt potenziell zu instabilem Lernen, beispielsweise durch oszillierende Strategien [23]. Die Entkopplung der Parameter $\theta^{-}$, die nach jeweils $n_{\theta}$ Zeitschritten durch $\theta$ ersetzt werden (Zeilen 12 und 13), wirkt dem durch die Stabilisierung der Zielgrößen entgegen. Zum Training der *$\mathcal{Q}$-Netze* werden bei DQN *mini-Batch* Optimierungsverfahren wie Adam verwendet (siehe Anhang A.0.2).

Für den DQN Basisalgorithmus existieren einige Erweiterungen vorgestellt. Drei gängige Erweiterungen, die in dieser Arbeit eine Rolle spielen sind namentlich *Prioritized Experience Replay* [24], *Double Q-Learning* [25] und *Dueling Q-Learning* [26]. Diese werden im Folgenden in verkürzter Form eingeführt.

*Prioritized Experience Replay* (PER) geht von der Annahme aus, dass Erfahrungs-Tupel mit hohem *Zeit-Differenz-Fehler* $|y_{\mathcal{Q}} - \mathcal{Q}|$ mehr Information tragen als Erfahrungs-Tupel mit niedrigem *Zeit-Differenz-Fehler*. Anstelle des gleich-verteilten Ziehens der Erfahrungs-Tupel (Zeile 10) werden durch PER Erfahrungen mit hohem *Zeit-Differenz-Fehler* priorisiert. Um sicherzustellen, dass die Priorisierung nicht zu einem Verlust an Diversität der Trainingsdaten führt, ist sie mit einer Stochastik versehen, wobei der Parameter $\alpha_{PER} \in \mathbb{R}_0^+$ den Einfluss der Priorisierung bestimmt. Zur weiteren Korrektur der hervorgerufenen Verzerrung der Erwartungswerte wird eine gewichtete *Importance Sampling* Methode mit einem weiteren Hyperparmeter $\beta_0 \in [0,1]$ genutzt (siehe [24] für Details).

Ein grundlegendes Problem des *Q-Learning Updates* ist eine systematische positive Verzerrung, im Folgenden *maximierungs-Bias* genannt, die dadurch entsteht, dass das Update auf einer Maximum-Suche über einer Menge von Schätzwerten (den Q-Werten im nächsten Zeitschritt) beruht. Bei der Umformung der DQN Zielgrößenberechnung (2.16) in

$$y_{\mathcal{Q}} \leftarrow r + \gamma \mathcal{Q}\big(\mathbf{s}', \arg\max_{a' \in \mathbf{A}} \mathcal{Q}(\mathbf{s}', a', \theta^-), \theta^-\big) \tag{2.17}$$

wird deutlich, dass sowohl die Bestimmung der Nachfolgeaktion $a'$, als auch die Berechnung des Erwartungswerts für $(\mathbf{s}', a')$ auf den $\mathcal{Q}$-Netz Parametern $\theta^-$ beruht. *Double Q-Learning* reduziert den skizzierten *maximierungs-Bias*, indem Parameter $\theta_{\mathrm{a}}$ zur Bestimmung von $a'$ von den Parametern $\theta_{\mathrm{b}}$, zur Schätzung der Q-Werte getrennt gelernt werden. Die Zielgrößen zum Update der Parameter $\theta_{\mathrm{a}}$ werden dann ermittelt durch

$$y_{\text{doubleQ}} \leftarrow r + \gamma \mathcal{Q}(\mathbf{s}', \arg\max_{a' \in A} \mathcal{Q}(\mathbf{s}', a', \theta_{\text{b}}), \theta_{\text{a}}). \tag{2.18}$$

Die Rolle von $\theta_{\text{a}}$ und $\theta_{\text{b}}$ wechselt während des Lernens, und wird für jedes Update zufällig festgelegt. Zur Erweiterung des DQN Algorithmus durch *double Q-Learning* werden die, wie oben beschrieben, entkoppelten Parameter $\theta$ und $\theta^-$ anstelle der getrennten Parameter $\theta_{\text{a}}$ und $\theta_{\text{b}}$ genutzt [25]. Die Zielgrößenberechnung aus (2.16) hat dann die Form

$$y_{\text{doubleDQN}} \leftarrow r + \gamma \mathcal{Q}(\mathbf{s}', \arg\max_{a' \in A} \mathcal{Q}(\mathbf{s}', a', \theta), \theta^-). \tag{2.19}$$

*Dueling Q-Learning* nutzt eine modifizierte Version des $\mathcal{Q}$-Netzes zum effizienteren Lernen. Die Modifikation basiert auf der Dekomposition $Q(s,a) = V(s) + \bar{A}(s,a)$ der Aktions-Bewertungsfunktion $Q$ in die bereits aus 2.2.2 bekannte Zustands-Bewertungsfunktion $V$ und die sogenannte *Advantage*-Funktion $\bar{A}$. Dies ermöglicht das Lernen der Bewertung $V(s)$ eines Zustands $s$ unabhängig von den Aktions-Bewertungen $Q(s,a)$ und soll so insbesondere bei Problemen mit großem Aktionsraum $A$ die Konvergenz begünstigen [26].

### 2.2.8 Einordnung der Lösungsverfahren

In den vorangegangenen Unterabschnitten wurde eine Auswahl von Methoden detailliert vorgestellt die auf Bewertungsfunktionen beruhen und direkt oder indirekt mit den in dieser Arbeit entwickelten Methoden verwandt sind. In diesem Unterabschnitt werden die Methoden zusammenfassend verglichen und weitere Methodenklassen des bestärkenden Lernens umrissen.
In Tabelle 2.1 sind die detailliert vorgestellten Methoden vergleichend aufgelistet. Während die Methoden des bestärkenden Lernens *online* lernen, nutzen Methoden der dynamischen Programmierung *a-priori* Modellwissen in Form

Tabelle 2.1: Einordnung der vorgestellten bewertungsbasierten Algorithmen

| Algorithmus | *Online* | Modellfrei | Approx. | *On-Policy* |
|---|---|---|---|---|
| Policy Iteration (Listing 1) | - | - | - | - |
| Value Iteration (Listing 2) | - | - | - | - |
| Backward DP (Listing 3) | - | - | - | - |
| Backward ADP (Listing 4) | * | - | + | - |
| Q-Learning (Listing 5) | + | + | - | - |
| SARSA | + | + | - | + |
| NFQ (Listing 6) | * | + | + | - |
| DQN (Listing 7) | + | + | + | - |

der Zustandsübergangsfunktion und Belohnungsfunktion um Entscheidungsprozesse *offline* zu lösen. Die vorgestellten Methoden des bestärkenden Lernens kommen ohne jegliches *a-priori* Modellwissen aus. Sonderfälle bezüglich dieser Kategorien sind *Backward ADP* und NFQ. *Backward ADP* nutzt die Zustandsübergangsfunktion $P$ zum Update der Zustands-Bewertungsfunktionen. Die Zustandsübergangsfunktion ist bei *Backward ADP* durch künstliche Neuronale Netze approximiert, die entweder *offline* auf Basis von vor-erzeugten Daten oder verfeinernd anhand von *online*-Daten trainiert werden können. Auch wenn *Backward ADP* grundlegend für die *offline* Anwendung ausgelegt ist, ist eine *online* Anwendung denkbar. NFQ ist ein Schema zum Training von künstlichen Neuronalen Netzen zur Approximation der Q-Funktion das sowohl *offline* mit gegebenem Datensatz als auch *online* in der beschriebenen inkrementellen Variante verwendet werden kann.
Grundlegende Algorithmen der dynamischen Programmierung und des bestärkenden Lernens, wie *Backward DP* oder *Q-Learning* basieren auf einer Repräsentation der beteiligten Bewertungsfunktionen in expliziter Form als Tabelle.

Approximative bewertungsbasierte Algorithmen wie *Backward ADP* oder *Deep Q-Networks* kombinieren die jeweiligen grundlegenden Algorithmen mit Mechanismen und Modellen zur approximativen Repräsentation der jeweiligen Bewertungsfunktionen und ermöglichen so die effiziente Anwendung bei hoch-dimensionalen oder kontinuierlichen Zustandsräumen.

Ein weiteres Merkmal zur Kategorisierung der vorgestellten bewertungsbasierten Ansätze ist die Art des Updates. Während das *on-Policy SARSA Update* Bewertungsfunktionen für die aktuell ausgeführte Strategie aktualisiert, ermöglichen *off-Policy* Updates das Lernen von Bewertungsfunktionen für die optimale Strategie aus Daten die einer abweichenden Strategie folgend erzeugt wurden.

Die bisher behandelten und die in dieser Arbeit entwickelten Algorithmen des bestärkenden Lernens sind modellfreie bewertungsbasierte Algorithmen. Sie basieren auf dem Lernen von Bewertungsfunktionen und die zum Lernen verwendeten Daten stammen aus der Interaktion mit dem Prozess. Ein Modell des Prozesses wird dabei nicht explizit repräsentiert. Eine gängige Taxonomie der Algorithmen des bestärkenden Lernens findet sich in Abbildung 2.2 (Nach David Silver[6]) in Form eines Venn-

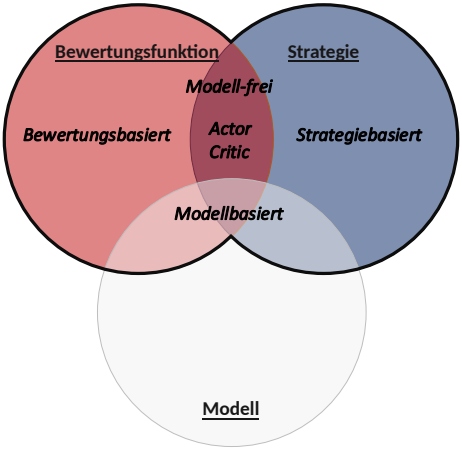



Abbildung 2.2: Taxonomie der Methoden des bestärkenden Lernens (Nach Silver[6])

[6] Kursfolien „Advanced Topics / Reinforcement Learning“, David Silver, University College London, 2015

Diagramms. Jede Basismenge des Diagramms steht dabei für einen Lösungsbestandteil, der durch den jeweiligen Algorithmus gelernt wird, oder im Fall des Modells in einigen Fällen auch *a-priori* gegeben ist.

*Strategiebasierte* Algorithmen optimieren direkt die Parameter $\theta$ einer üblicherweise stochastischen und bezüglich $\theta$ differenzierbaren Strategie $\pi(a,s,\theta) = Pr(a|s,\theta)$. Aktuelle Beispiele für *strategiebasierte* Verfahren sind *Trust Region Policy Optimization* [27] und *Proximal Policy Optimization* [28]. Durch das direkte Lernen der Strategie ist keine Maximum-Suche über die Aktionen $A$ notwendig. Entgegen den vorgestellten Methoden des bewertungsbasierten Lernens, können strategiebasierte Methoden somit ohne weiteres für Probleme mit kontinuierlichen Aktionsbeschreibungen $\mathbf{a} \in \mathbb{R}^n$ angewandt werden. Des Weiteren zeichnet sich *strategiebasiertes* Lernen üblicherweise durch stabiles Konvergenzverhalten aus, während beim *bewertungsbasierten* Lernen kleine Änderungen der Bewertungsfunktionen zu großen Änderungen der zugehörigen *greedy*-Strategie führen können. Ein Nachteil *strategiebasierter* Algorithmen ist jedoch, dass entgegen den *bewertungsbasierten* Methoden in jedem Lernschritt die Parameter $\theta$ der parametrisierten Strategie $\pi(a,s,\theta)$ nur durch aktuelle Daten optimiert werden können, Updates also notwendigerweise *on-Policy* erfolgen. Dies macht das Lernen aus Prozessdaten, die unter alten Formen der Strategie erzeugt wurden (*Experience Replay*) unmöglich, was wiederum in vielen Fällen zu einer geringeren Dateneffizienz der Algorithmen führt [29].

*Actor Critic* Methoden Lernen sowohl ein Modell der Strategie als auch ein Modell der Bewertungsfunktion und stellen somit eine Mischform dar. Ziel ist dabei, die Vorteile beider Formen des bestärkenden Lernens zu vereinen. Konkret vereinen einige Algorithmen dieser Klasse die Fähigkeit Strategien mit kontinuierlichen Aktionen zu optimieren mit der Möglichkeit aus *off-Policy* Daten zu lernen. Aktuelle Beispiele sind *Actor Critic with Experience Replay* [30] und *Soft Actor Critic* [31].

*Modellbasierte* Algorithmen des bestärkenden Lernens nutzen während des Lernens ein Zustandsübergangsmodell der Umgebung. Das Umgebungsmodell wird dabei entweder parallel zur Bewertungsfunktion beziehungsweise der Strategie *online* gelernt, oder bereits im Vorfeld gelernt und basierend auf *online* Daten verfeinert. Eine Ausnahme stellt beispielsweise der *AlphaGo* Ansatz [32] dar, bei dem ein *a-priori* Umgebungsmodell zur Planung der Züge mit *modellfreiem* Lernen zur Einschätzung der langfristigen Effekte verschiedener Züge kombiniert wird. Umgebungsmodelle können beim bestärkenden Lernen beispielsweise mit dem Ziel genutzt werden Realdaten zu ergänzen und so die Dateneffizienz zu erhöhen [33, 34], oder um Stabilitätskriterien auf explorative Lernstrategien bei sicherheitskritischen Anwendungen anwenden zu können [35]. Die Genauigkeit des Umgebungsmodells hat dabei einen wesentlichen Einfluss auf die Qualität der Lernergebnisse. Wird das Modell zum Lernen über mehrere Zeitschritte ausgerollt, können selbst geringe Ungenauigkeiten zu hohen akkumulierten Fehlern führen [36].
Neben diesen allgemeinen Algorithmen des bestärkenden Lernens existiert eine Vielzahl von Methoden zur Behandlung spezifischer Charakteristika der Entscheidungsprozesse. Solche erweiterten Ansätze, die für die vorliegende Arbeit von besonderer Bedeutung sind, werden in Abschnitt 2.3 behandelt.

## 2.3 Erweiterte Ansätze des bestärkenden Lernens

### 2.3.1 Partielle Beobachtbarkeit

Die vorangehend beschriebenen Methoden der dynamischen Programmierung und des bestärkenden Lernens gehen von einer vollständigen Beobachtbarkeit

des Prozesses im Sinne der Problemformulierung als Markov-Entscheidungsprozess (siehe 2.2.1) aus. Die *Markov-Eigenschaft* impliziert dabei, dass in jedem Zeitschritt eine Zustandsbeschreibung gegeben sein muss, auf deren Basis die optimale Aktion bezüglich des Entscheidungsprozesses bestimmt werden kann. In vielen Anwendungsfällen, wie der *online*-Optimierung von Fertigungsprozessen, ist der Zustand $s$ jedoch nicht direkt zugänglich. Stattdessen sind beobachtbare Größen $o \in \Lambda$ gegeben, die nicht vollständig auf den Zustand im Sinne der *Markov-Eigenschaft* schließen lassen. Formal sind solche Probleme als *partiell beobachtbare Markov-Entscheidungsprozesse* (POMDP) $(S, A, P, R, \gamma, O, \Lambda)$ darstellbar, wobei die Menge $\Lambda$ die beobachtbaren Größen umfasst und $O(o_t, s_{t+1}, a_t) = Pr(o_t | s_{t+1}, a_t)$ die Beobachtungswahrscheinlichkeit von $o_t$, bedingt durch $s_{t+1}$ und $a_t$ angibt. Im Folgenden werden verschiedene Ansätze zum Umgang mit partiell Beobachtbaren Szenarien skizziert.

Ein modellbasierter Ansatz zur Lösung von POMDPs ist das Erlernen von Bewertungsfunktionen für sogenannte *Belief States*. *Belief States* bilden die Wahrscheinlichkeitsverteilung $Pr(s_t | o_0, a_0, ..., o_t, a_{t-1})$ der aktuellen Prozesszustände $s_t$ in Abhängigkeit von den bisher ausgeführten Aktionen und beobachteten Größen ab [37]. Die Menge der bisher ausgeführten Aktionen und beobachteten Größen werden im Folgenden als historische Prozessdaten bezeichnet. Ein weiterer modellbasierter Ansatz zum Umgang mit partieller Beobachtbarkeit, der beispielsweise in [3] verfolgt wird, ist die Nutzung eines Beobachtermodells zur Punktschätzung des aktuellen Zustands $\tilde{\mathbf{s}} \approx \mathbf{s}$ auf Basis der aktuell beobachteten Größen oder der historischen Prozessdaten. Dabei wird davon ausgegangen, dass der Entscheidungsprozess auch für die geschätzten Zustände $\tilde{\mathbf{s}}$ die Markov-Bedingung erfüllt. Diese Annahme wird im Folgenden als *Markov-Annahme* bezeichnet.

Ein modellfreier Ansatz ist die Berücksichtigung der historischen Prozessdaten bei der Approximation der Bewertungsfunktion. Im einfachen Fällen reicht es aus, für die Zusammenfassung der letzten $n$ beobachteten Größen die *Markov-Annahme* zu treffen, und die die Konkatenation der letzten $n$ beobachteten Größen $\tilde{\mathbf{s}} = \mathbf{o}_t \frown \mathbf{o}_{t-1} \frown ... \frown \mathbf{o}_{t-n+1}$ als Zustandsvektor $\tilde{\mathbf{s}}$ zu betrachten.

Dieser Ansatz wird beispielsweise in [23] mit $n = 4$ zum Lernen von Computerspiel-Strategien verfolgt, wobei sich $\mathbf{o}_t$ aus vorverarbeiteten Farbinformationen eines Bildschirminhaltes zusammensetzt. Für Anwendungsfälle bei denen historische Prozessdaten über einen längeren Zeitraum eine Rolle spielen, werden häufig Sequenz-Modelle, wie *rekurrente künstliche Neuronale Netze*, als Bewertungsfunktion verwendet um den aktuellen Zustand implizit aus der Sequenz der historischen Prozessdaten abzuleiten [38–40]. Bei dem in dieser Arbeit entwickelten BFHNQ Algorithmus wird mit dem gleichen Ziel ein Ansatz verfolgt, bei dem für Zeitschritt-abhängige Q-Funktionen die gesamte Sequenz der historischen Prozessdaten als Zustandsbeschreibung genutzt wird (siehe 4.2).

### 2.3.2 Multikriterielles bestärkendes Lernen

Die skalare Belohnungsfunktion setzt sich bei einigen Anwendungen aus mehreren, oft gegenläufigen, Belohnungskriterien zusammen, deren relative Gewichtung von einer situationsbedingten Konfiguration abhängt. Methoden des multikriteriellen bestärkenden Lernens [41] können in zwei Klassen eingeteilt werden: (a) Methoden zum Lernen einzelner Regelungsstrategien haben zum Ziel, für eine gegebene Gewichtung der Kriterien eine optimale Regelungsstrategie zu finden. (b) Methoden zum Lernen vielfältiger Regelungsstrategien haben zum Ziel die *Pareto-Optima* der Regelungsstrategien (siehe 4.5) im Raum der Belohnungskriterien zu approximieren.
In Abschnitt 4.5 wird multikriterielles bestärkendes Lernen genutzt, um zu ermöglichen, dass unter einer Konfiguration gelerntes Prozesswissen auf neue Konfigurationen übertragen werden kann. Verwandte Arbeiten, die multikriterielles bestärkendes Lernen auf ähnliche Weiße nutzen sind [42–44]. In [42] wird der *R-Learning* Algorithmus (siehe 2.2.1) mit dem Speichern mehrerer separat gelernter Bewertungsfunktionen kombiniert, um effizient unter sich über

die Zeit verändernden Konfigurationen zu lernen. In [43] wird ein multikriterieller Ansatz zum bestärkenden Lernen autonomer Überholmanöver im Autoverkehr vorgestellt. Hierfür werden Q-Funktionen für sieben Kriterien separat gelernt und in der Anwendung durch einen Planungsalgorithmus gewichtet gemittelt. In [44] wird eine Methode zum Transfer von gelerntem Prozesswissen in einer multikriteriellen multi-Agenten Umgebung vorgestellt und anhand eines *smart-Grid* Optimierungsproblems untersucht [44].

### 2.3.3 Generalisierung über Zielbeschreibungen

Bei einigen Anwendungen ist es möglich, neben dem Zustand $s$ auch eine Beschreibung $\mathbf{g}$ des Zielzustands $g$, im Folgenden auch als Ziel bezeichnet, anzugeben. Diese Zielbeschreibungen werden auf verschiedene Arten beim bestärkenden Lernen genutzt. Der in dieser Arbeit entwickelte Ansatz zum bestärkenden Lernen bei mehreren äquivalenten Zielen (siehe 5.2.2) ist eng verwandt mit einer Reihe aufeinander aufbauender Arbeiten [45–47], die im Folgenden vorgestellt werden.

In [45] stellen Sutton et al. einen systematischen Ansatz vor, um anstelle von Bewertungsfunktionen, die bezüglich des Gesamtziels des Agenten definiert sind, mehrere sogenannte *General Value Functions* als Bausteine (orig. *Deamons*) einer speziellen, *Horde* genannten, Architektur zu verwenden. *General Value Functions* bilden die Erwartungswerte bezüglich spezifischer, dem Lernziel angepasster, pseudo-Belohnungsfunktionen ab. Genutzt werden kann dieser Ansatz um Wissen bezüglich des Entscheidungsprozesses durch die Gestaltung semantischer pseudo-Belohnungsfunktionen als Wissensrepräsentation zu modellieren. Diese Wissensrepräsentation kann beispielsweise genutzt werden, um Systemverhalten vorauszusagen [45], Zustandsbeschreibungen anzureichern [48], oder um bestärkendes Lernen auf hierarchisch angeordneten Zeitebenen durchzuführen (vgl. [16] S. 461-464).

Der Idee von *General Value Functions* folgen Schaul et al. [46] mit ihrem Ansatz, indem sie pro Ziel $g$ eine pseudo-Belohnungsfunktion $Q_g(s,a)$ definieren. Die *Universal Function Approximators* genannten approximierten Bewertungsfunktionen $\mathcal{Q}(\mathbf{s},a,\mathbf{g},\theta)$ nehmen Zielbeschreibungen $\mathbf{g}$ als Funktionsargument auf und generalisieren so über die pseudo-Belohnungsfunktionen. Hierin unterscheidet sich der Ansatz von [45], wo eine Bewertungsfunktion pro pseudo-Belohnungsfunktion gebildet wird.
Andrychowicz et al. bauen in ihrem *Hindsight Experience Replay* genannten Ansatz [47] wiederum auf *General Value Functions* auf. *Hindsight Experience Replay* ist eine Methode zur *Augmentierung*[7] des *Replay Memory* $\mathcal{D}$ durch hypothetische Erfahrungen, die unter der Annahme erzeugt werden, dass während des Lernens anstelle des verfolgten Ziels $g$ ein hypothetisches anderes Ziel $\hat{g}$ verfolgt wurde.
Eine weitere Reihe von Arbeiten, bei denen die Definition von Zielen eine große Rolle spielt ist das *hierarchische bestärkende Lernen*. Anders als in den bisher besprochenen Arbeiten, sowie in der vorliegenden Arbeit, werden in diesen hierarchischen Ansätzen durch den Agenten auf einer Planungsebene Teilziele identifiziert, die für das Erreichen des Gesamtziels vorteilhaft sind, während auf einer Ausführungsebene gelernt wird diese Teilziele zu erreichen. Bedeutende Ansätze die diesem Muster folgen werden beispielsweise in [49, 50] vorgestellt.
Arbeiten bei denen, wie bei der in dieser Arbeit entwickelten und in 5.2.2 vorgestellten Methode, generalisierte Belohnungsfunktionen gelernt werden, um Entscheidungsprozesse mit mehreren äquivalenten Zielen effizient zu lösen wurden in der Literaturrecherche nicht ausfindig gemacht.

[7] Erweiterung eines Datensatzes um synthetisch erzeugte Daten

### 2.3.4 Dünn besetzte Belohnungssignale

Die zentrale Herausforderung beim Lernen der Bewertungsfunktionen aus Erfahrungen ist die korrekte Identifikation von Aktionen, die zu später beobachteten Belohnungssignalen geführt haben. Diese, *Credit Assignment Problem* genannte, Herausforderung wird erheblich erschwert, wenn von dem Nullwert abweichende Belohnungssignale nur selten auftreten. Gleichzeitig ist es bei einigen Anwendungen schwierig den erreichten Zustand vor dem Ende der Episode zu bewerten. Unter anderem deshalb existieren einige Ansätze zum Umgang mit dünn besetzten Belohnungssignalen.

Ein verbreiteter Ansatz ist die Nutzung von Heuristiken zur Bewertung von Zwischenzuständen durch sogenanntes *Reward Shaping*, im Weiteren als Umformung der Belohnungsfunktion bezeichnet. Dies hat in vielen Fällen den Nachteil, dass dadurch Erwartungswerte verfälscht werden und für die umgeformte Belohnungsfunktion optimale Strategien kein Optimum bezüglich der Belohnungsfunktion vor der Umformung darstellen. In [51] wird eine *Potential Based Reward Shaping* genannte Form der Umformung vorgestellt, für die garantiert ist, dass optimale Strategien $\pi^*$ und Strategien nahe $\pi^*$ invariant bezüglich der Umformung sind. *Potential Based Reward Shaping* wird in in 5.2.1 formal eingeführt.

Ein weiterer Ansatz zum Umgang mit dünn besetzten Belohnungssignalen im Fall von Problemen mit beschreibbaren Zielen $g$ ist das bereits im vorangegangenen Unterabschnitt vorgestellte *Hindsight Experience Replay* [47]. Durch die beschriebene *Augmentierungs*-Technik wird hier erreicht, dass über Ziele generalisierende approximative Bewertungsfunktionen $\mathcal{Q}(\mathbf{s}, a, \mathbf{g})$ auch im Fall von dünn besetzten Belohnungssignalen effizient gelernt werden können.

## 2.4 Entscheidungsoptimierung für Fertigungsprozesse

In diesem Abschnitt werden Arbeiten vorgestellt, die mit unterschiedlichen Zielstellungen Methoden zur Entscheidungsoptimierung und optimalen Regelung im Kontext der Fertigung anwenden. In 2.4.1 werden Methoden der *modellprädiktiven Regelung* und Methoden des bestärkenden Lernens zur optimalen Regelung und Optimierung von Regelstrategien in dem Kontext der Fertigungsprozesse behandelt. In 2.4.2 werden verwandte Arbeiten vorgestellt, die dem Bereich *optimal operational Control* zuzuordnen sind. Abschließend werden in 2.4.3 und 2.4.4 Arbeiten vorgestellt, die Aufgrund der behandelten Anwendungsfälle mit der vorliegenden Arbeit verwandt sind.

### 2.4.1 Optimale Regelung

Traditionell werden im Bereich der optimalen Regelung modellbasierte Verfahren, insbesondere Verfahren der linearen *modellprädiktiven Regelung* untersucht und angewandt. Eine umfassende Zusammenfassung von Ausprägungen der *modellprädiktiven Regelung* sowie von Anwendungen in der optimalen Prozessregelung findet sich in [52]. Prozessmodelle in der *modellprädiktiven Regelung* werden üblicherweise durch lineare Methoden zur Systemidentifikation ermittelt. Ein aktueller Forschungsschwerpunkt ist die Entwicklung und Untersuchung von Methoden zur nichtlinearen *modellprädiktiven Regelung* [53]. Bei der *modellprädiktiven Regelung* wird das Prozessmodell online zur Planung der nächsten Regelschritte genutzt. Hierdurch sind die Rechenkosten während der optimalen Regelung direkt abhängig von den Kosten des Prozessmodells. Dies führt dazu, dass *modellprädiktive Regelung* in einigen Fällen nicht oder nur in sehr eingeschränkter Form eingesetzt werden kann, da die Optimierung nicht in der erforderlichen Zeit durchführbar ist. Ansätze, um dem zu begegnen sind offline Vorausberechnungen der optimalen Regelung

[54] oder die Verwendung von künstlichen Neuronalen Netzen zur Approximation des Prozessmodells [55, 56].
Modellfreies bestärkendes Lernen wie es in dieser Arbeit untersucht wird und *modellprädiktive Regelung* werden in einer Reihe aktueller Arbeiten vergleichend diskutiert [57, 58]. Eine inhärente Eigenschaft von Methoden des modellfreien bestärkenden Lernens ist die Fähigkeit Prozessverhalten zu adaptieren [59], während existierende Methoden zur adaptiven *modellprädiktiven Regelung* sich laut Görges [57] üblicherweise auf die Robustifizierung der Regelungsstrategie beschränken. Durch die rekursive Erwartungswertformulierung ist der Zeithorizont über den die zukünftig erwarteten Belohnungen (bzw. Kosten) betrachtet werden beim bestärkenden Lernen unbegrenzt, während dieser bei der *modellprädiktiven Regelung* auf den Prädiktionshorizont beschränkt ist. Shin et al. [58] betonen den oben besprochenen Vorteil des bestärkenden Lernens, ohne umfangreiche Berechnungen während der Regelung auszukommen, während bei der *modellprädiktiven Regelung* in jedem Zeitschritt drei Optimierungsprobleme sequenziell gelöst werden müssen. Vorteile der *modellprädiktiven Regelung* gegenüber modellfreiem bestärkendem Lernen sehen sowohl Görges, als auch Shin et al. vor allem in der Möglichkeit Nebenbedingungen in Form von Einschränkungen der erlaubten Systemzustände zu definieren und in der Möglichkeit Aussagen bezüglich der Robustheit und der Stabilität der resultierenden optimalen Regelung zu treffen [57, 58]. In [60] werden nicht-lineare *modellprädiktive Regelung* und ein modellfreier *Actor Critic* Ansatz vergleichend zur optimalen Regelung eines invertierten Pendels untersucht. Dabei wird durch Variation der Modell-Genauigkeit gezeigt, dass modellfreies bestärkendes Lernen ab einem gewissen Break-even-Punkt der *modellprädiktiven Regelung* überlegen ist.
Einige Arbeiten haben zum Ziel die Vorteile von modellfreiem bestärkenden Lernen und der *modellprädiktiven Regelung* durch hybride Ansätze zu vereinen. So werden etwa in [61] Ansätze zur Nutzung von modellfreiem bestärkendem Lernen zur Kompensierung von Modell-Ungenauigkeiten bei der *modellprädiktiven Regelung* vorgestellt.

Bestärkendes Lernen stellt eine Möglichkeit dar stochastische Systeme auf adaptive Art optimal zu regeln [59]. Aktuelle Arbeiten zu Algorithmen des bestärkenden Lernens und der dynamischen Programmierung für die optimale Prozessregelung werden in [62] verglichen und kategorisiert. Dabei werden Arbeiten aus drei Kategorien untersucht: (a) Generelle Arbeiten zur optimalen Regelung, (b) Arbeiten zur optimalen Regelung bei gegebenen Sollwerten und (c) Arbeiten zur optimalen $\mathcal{H}_\infty$ Regelung. Ein Großteil der untersuchten Arbeiten behandelt modellbasiertes bestärkendes Lernen zur Optimierung initial gegebener Regelungen. Arbeiten zur optimalen Regelung mittels modellfreiem bestärkendem Lernen die vorgestellt werden sind unter anderen [63, 64]. In [63] wird ein *Q-Learning* ähnliches Verfahren vorgestellt um mittels modellfreiem bestärkendem Lernen Optimierungsprobleme der linear-quadratischen Regelung (LQR) zu lösen. Der vorgestellte Ansatz integriert die Gütefunktion der LQR in die Bellman Gleichung (2.3) und leitet davon einen modellfreien Ansatz ab der auf lineare, deterministische, partiell beobachtbare Systeme anwendbar ist. In [64] wird auf ähnliche Weise die Gütefunktion der linear-quadratischen Folgeregelung in die Bellman Gleichung integriert.
Weitere Arbeiten bei denen bestärkendes Lernen zur optimalen Regelung beziehungsweise zur Optimierung von Regelungsstrategien angewandt wird sind [65–67]. In [65] wird ein *Actor Critic* Ansatz zur optimalen Regelung der Laserleistung bei einem Laserschweißprozess mit vorgegebener Schweißtiefe vorgestellt. In [66] wird ein strategiebasierter Ansatz (*Deep Deterministic Policy Gradient*) auf einen simulierten Polymerisationsprozess mit simuliertem Messrauschen angewandt. In [67] wird der *Deep Q-Networks* Algorithmus in Kombination mit künstlichen neuronalen Netzen mit Faltungsschichten zur Optimierung eines mittels finite Elemente Methode simulierten Freiform-Präge-Prozesses verwendet.

### 2.4.2 Operational Control

Ein weiterer Anwendungsbereich des bestärkenden Lernens im Bereich der Optimierung von Fertigungsprozessen ist die sogenannte *optimal operational Control*. Diese ist als zwei-Ebenen Regelungssystem definiert. Auf der Geräteebene (engl. *Device Layer*) findet optimale Regelung unter Sollwertvorgabe statt. Auf dem *operational Layer* werden anhand vorgegebener operationaler Zielgrößen (bspw. bezüglich der Produktqualität und des Materialverbrauchs) Sollwertvorgaben für die Regelung auf Geräteebene bestimmt [68]. Aktuelle Beispiele für die Verwendung von modellfreiem bestärkendem Lernen im Rahmen der *optimal operational Control* sind [69, 70]. In [69] wird ein *PI-Regler* auf Geräteebene mit einem modellfreien *Actor Critic* Verfahren auf dem *Operational Layer* kombiniert, um nichtlineare Schwimmaufbereitungsprozesse zu regeln. In [70] wird modellfreies bestärkendes Lernen (mittels *Q-Learning*) auf beiden Regelungsebenen angewandt, um einen linearisierten Eindickungsprozess zu optimieren.
Einige Arbeiten fokussieren sich auf die Optimierung auf dem *Operational Layer* durch modellfreies bestärkendes Lernen. Aktuelle Beispiele hierfür sind [71] und [72]. In [71] wird die Anwendung von tiefem bestärkendem Lernen zur optimal adaptiven Regelung der Prozesslast bei stark schwankenden Energiepreisen untersucht. In [72] werden Verfahren des bestärkenden Lernens zur optimalen Ressourcendisposition in Produktionsumgebungen angewendet und untersucht.

### 2.4.3 Tiefziehprozess

In der vorliegenden Arbeit dient der Tiefziehprozess als Anwendungsbeispiel um die vorgestellten Methoden zu instanziieren und evaluieren. Beim Tiefziehen, in der Form, in der es hier betrachtet wird, wird ein Blechzuschnitt durch einen *Ziehstempel* in das innere einer *Ziehmatrize* gedrückt, mit dem Ziel den Blechzuschnitt in einen Hohlkörper umzuformen. *Niederhalter* fixieren dabei

das Blech zwischen Stempel und Matrize. Neben vorgegebenen Parametern, wie etwa dem verwendeten Material oder der Schmierung des Prozesses, hat die gewählte *Niederhaltekraft* erheblichen Einfluss auf die Qualität des resultierenden Werkstücks. Der Einfluss der zeitlichen und räumlichen Variation der Niederhaltekraft auf das Prozessergebnis wird unter anderem in [73], [74] und [75] experimentell untersucht.

Verwandte Arbeiten die sich mit der optimalen Regelung des Tiefziehprozesses beschäftigen sind [2, 76–79]. Die bereits in Abschnitt 2.1 eingeführte Arbeit [2] behandelt verschiedene Ansätze des *approximate Dynamic Programming* zur Optimierung der zeitlichen Variation der Niederhaltekraft basierend auf vor-generierten Prozessdaten. Dabei wird unter anderem der in 2.2.4 beschriebene *Backward Approximate Dynamic Programming* Algorithmus vorgestellt. Endelt et al. [76, 77] stellen über mehrere Arbeiten hinweg verschiedene Methoden zur Regelung von Tiefziehprozessen vor. Betrachtete Prozessparameter sind dabei, neben der Niederhaltekraft, Druckwerte von über die Niederhaltefläche verteilt aufgebrachten hydraulischen Kissen. In [76] wird ein Verfahren zur optimalen Regelung des Tiefziehprozesses vorgestellt. Ziel der optimalen Regelung ist die Minimierung der Abweichung des Flansch-Einzugs von einer Referenztrajektorie pro Zeitschritt. Die Referenztrajektorie wird im Vorfeld manuell ermittelt und es wird angenommen, dass der Prozess für diese robust ist. Der Flansch-Einzug an verschiedenen Messpunkten wird als beobachtbare Größe für die optimale Prozessregelung bestimmt. Die Lösung des Regelungsproblems basiert auf einem linearen Zustandsraummodell mit klassischer Zustandsregelung. Optimale Verstärkungsfaktoren der linearen Regelung werden mittels der Methode der kleinsten Quadrate mit nichtlinearer Modellfunktion basierend auf FEM Simulationen ermittelt. Geregelt wird hier unabhängig pro Tiefziehvorgang. Für einen vergleichbaren Anwendungsfall werden durch Fischer et al. in [78] weitere Methoden zur optimalen Regelung vorgestellt, die auf der Linearisierung des Modells am Arbeitspunkt beruhen. Um Wissen über korreliertes Prozessverhalten aufeinanderfolgender Tiefziehvorgänge

(beispielsweise durch Wärmeentwicklung, durch Reibung oder durch Werkzeugverschleiß) bei der optimalen Regelung zu nutzen wird in [77] ein äußerer Regelkreislauf mittels iterativ lernender Regelung und Filtermethoden als Ergänzung des Ansatzes aus [76] vorgeschlagen. Beobachtbare Größen sowie der innere Regelkreislauf sind dabei weitgehend identisch zu [76]. Auch hier werden Verstärkungsfaktoren der iterativ lernenden Regelung mittels der Methode der kleinsten Quadrate mit nichtlinearer Modellfunktion ermittelt. Evaluiert wird der Ansatz anhand von Tiefziehsimulationen mit systematischer Variation des Reibungskoeffizienten und einiger Materialmodellparameter. Eine Gemeinsamkeit mit der vorliegenden Arbeit ist neben dem Anwendungsfall, dass durch Endelt et al. unter anderem FEM Simulationen auf interaktive Art zur Evaluation der Ansätze verwendet werden. Neben der Verwendung eines Prozessmodells sind die Hauptunterschiede zur vorliegenden Arbeit, dass die zu optimierende Größe pro Zeitschritt in Form der Abweichung von einer Referenztrajektorie gegeben ist und Zusammenhänge zwischen der zu minimierenden Größe und den zu optimierenden Prozessparametern explizit modelliert sind.

Guo und Yu wenden in [79] tiefes bestärkendes Lernen zur optimalen Regelung des Tiefziehprozesses an. Dabei bauen sie auf in [80] veröffentlichten Teilen der vorliegenden Arbeit auf, indem sie die Problemformulierung, unter anderem in Form der *Belohnungsfunktion*, aus dieser übernehmen. Durch Verwendung eines *Actor Critic* Ansatzes werden kontinuierliche Niederhaltekräfte optimiert. Guo und Yu gehen hierbei von einer direkten Beobachtbarkeit der *Von-Mises-Vergleichsspannungen* aus und nutzen diese als Zustandsbeschreibung. Da in [79] ein abweichendes, nicht veröffentlichtes Simulationsmodell verwendet wird und der Grad der Beobachtbarkeit des Prozesses stark abweicht ist ein direkter Vergleich der Ergebnisse aus [79] mit den Ergebnissen der vorliegenden Arbeit leider nicht möglich.

### 2.4.4 Struktur-geleitete Prozesspfadoptimierung

Ziel der Struktur geleiteten Prozesspfadoptimierung, wie sie in Kapitel 5 vorgestellt wird ist die Optimierung Material-Struktur-verändernder Prozesse. Die vorgestellte Arbeit ergänzt dabei aktuelle Arbeiten aus der Materialwissenschaft zur Invertierung von Struktur-Eigenschafts-Abbildungen. Da es sich bei Struktur-Eigenschafts-Abbildungen meist um $n$ zu 1 Abbildungen handelt, ermitteln einige Verfahren mehrere äquivalente Strukturen die die gewünschten Zieleigenschaften aufweisen. Aktuelle Beispiele hierfür sind [81, 82]. In beiden Fällen werden Suchverfahren mit überwacht gelernten Filtermethoden zur Invertierung von Struktur-Eigenschaftsabbildungen kombiniert. Verschiedene Arbeiten die sich mit der Optimierung von Prozessparametern oder Prozesspfaden zur Erreichung gegebener Ziel-Mikrostrukturen [83–89] beschäftigen werden im Folgenden eingeführt.

Shaffer et al. [83] stellen ein sogenanntes *Texture Evolution Network* vor. Dieses kann als gerichteter Graph gesehen werden, wobei Struktur-Repräsentationen (kristallografische Texturen) als Knoten repräsentiert sind, die durch Prozessschritte repräsentierende gerichtete Kanten verbunden sind. Der Graph wird mittels *a priori* abgetasteten Prozessdaten erstellt und kann durch eine Vorwärtsabbildung von Strukturen auf Eigenschaften in den Eigenschaftsraum transferiert werden. Für eine gegebene Startstruktur und gegebene Zieleigenschaften besteht die Prozesspfadoptimierung dann aus der Suche eines Pfades im transferierten *Texture Evolution Network* von dem Startknoten zu einem im Zieleigenschaftsbereich liegenden Endknoten.

Li et al. nutzen in [84] und [85] Strukturevolutionsmodelle für kristallographische Texturen (nach Bunge [90]) um Pfade unterschiedlich parametrisierter Deformationsprozesse durch Stromlinien im Raum der Spektral-Koeffizienten (siehe Anhang A.0.1) zu repräsentieren. Die Gesamtmenge dieser Pfade bildet ein Netz. Wie in [83] können diese Netze genutzt werden, um in der Anwendung durch Pfadsuche einen Prozesspfad von der Startstruktur zur gewünschten Zielstruktur zu ermitteln. Betrachtete Deformationsoperationen sind dabei

die einachsige Dehnung und das Walzen des Materials. Die beiden Arbeiten von Li et al. unterscheiden sich im Wesentlichen bezüglich des Kristallsystems des betrachteten Materials.

In [86] werden für unterschiedliche Deformationsprozesse mittels Dimensionsreduktion durch *Hauptkomponentenanalyse* sogenannte Prozessebenen ("process planes") im Raum der Strukturen ermittelt. Für eine gegebene Zielmikrostruktur wird anschließend die Prozessebene gesucht, die diese am genauesten repräsentieren kann. Der zugehörige Prozess wird dann als der Prozess angenommen, der die Zielmikrostruktur am besten erreichen kann. In [87] wird diese Methode ergänzt, indem Prozessebenen für Sequenzen von bis zu drei Prozessen erstellt werden.

Dagegen nutzen Sundar et al. [88] *Variational Autoencoder* um Mikrostrukturen nicht-linear in einen niedrig-dimensionalen Raum abzubilden. *A priori* abgetastete Prozesspfade werden zusammen mit auf den Pfaden liegenden Mikrostrukturen in expliziter Form abgespeichert. Für eine gegebene Zielstruktur wird in der Anwendung die Datenbank nach der ähnlichsten bekannten Struktur gesucht und der zugehörige Prozesspfad als Lösung genutzt. Zum Vergleich der Strukturen wird eine Distanzfunktion im dimensionsreduzierten Raum genutzt. Angewandt wird die vorgestellte Methode auf Daten aus simulierten Prozessen zur Dehnung, zum Walzen und zur Scherung von Stahltexturen.

Ein Ansatz der mittels sogenanntem *active Learning* das Prozessmodell während der Optimierung abtastet wird durch Tran et al. [89] vorgeschlagen. Eine Kombination aus Bayes Optimierung und kinetischen Monte Carlo Simulationen werden hier zur Prozessoptimierung mit der Zieldefinition im Strukturraum genutzt. Die Methode wird anhand eines Schweißprozesses und anhand der Optimierung der Temperatur beim Kornwachstum evaluiert.

Der Großteil der oben aufgelisteten Arbeiten basiert dabei auf Methoden die entweder direkt in einer Menge vorberechneter Prozesspfade nach einer Lösung suchen [83, 86–88] oder eine generalisierende Struktur von vorberechneten Prozesspfaden ableiten und diese als Grundlage für eine Lösungssuche

verwenden [84, 85]. Die einzige unter den identifizierten Arbeiten, die zur Optimierungszeit den Prozess abtastet ist [89]. Ein Großteil der besprochenen Arbeiten beschäftigt sich mit der Optimierung von Prozessen mit zeitunabhängigen Parametern [86, 89] (Dies entspricht Prozesspfaden der Länge 1) oder mit der Optimierung kurzer Prozesspfade (bzw. Prozesssequenzen) bestehend aus maximal sechs Prozessschritten [87, 88].

# 3 Forschungsaufgabe

Ziel der Arbeit ist die Entwicklung und Untersuchung von Algorithmen zur *online* Ermittlung von optimalen Regelungsstrategien für Fertigungsprozesse. Wie einleitend in Kapitel 1 beschrieben, stellt die Findung und Optimierung von Regelungsstrategien eine Form der optimalen Regelung dar. Zur Formulierung der Forschungsaufgabe werden im Folgenden die in Kapitel 2 vorgestellten allgemeinen Lösungsansätze zur modellbasierten optimalen Regelung und des bestärkenden Lernens im Kontext der Fragestellung diskutiert. Anschließend werden die in der Arbeit behandelten konkreten Aufgabenklassen eingeordnet und die in Kapitel 2.3 eigeführten erweiterten Ansätze des bestärkenden Lernens eingeordnet. Daraus wird der Lösungsbedarf identifiziert, woraus wiederum die notwendigen Entwicklungs- und Untersuchungsgegenstände der Dissertation identifiziert werden.

Ein in der Praxis weit verbreitetes Verfahren zur optimalen Regelung ist die *modellprädiktive Regelung*. Daneben werden hauptsächlich in der Forschung modellbasierte Verfahren des bestärkenden Lernens und der dynamischen Programmierung zur optimalen Regelung behandelt. Exemplarische modellbasierte Verfahren und Forschungsansätze der *modellprädiktiven Regelung* wurden in 2.4.1 vorgestellt. Der Unterabschnitt 2.2.8 gibt einen Überblick zu modellbasierten Verfahren des bestärkenden Lernens und der dynamischen Programmierung. Die Anwendung modellbasierter Verfahren setzt ein gegebenes Prozessmodell sowie eine gegebene Belohnungsfunktion voraus. Zweiteres stellt im Kontext der Prozessoptimierung für gewöhnlich kein Problem dar, da es sich bei der Belohnungsfunktion hier in aller Regel um eine zum Zweck der

Optimierung definierte Funktion handelt, die folglich bekannt ist. Die Forderung nach einem expliziten Modell des Prozesses stellt jedoch eine hohe Hürde dar. Falls überhaupt möglich, ergibt sich ein hoher Aufwand bei der Erstellung des Modells, das gleichzeitig den Prozess ausreichend genau abbilden muss und rechnerisch ausreichend performant sein muss. Im Fall der modellprädiktiven Regelung, und Teilen des modellbasierten bestärkenden Lernens und des *adaptive dynamic programming* wird das Modell *online* zur Simulation der Auswirkung verschiedener Aktionen auf Folgezustände innerhalb eines definierten Prädiktionshorizonts verwendet. Dadurch sind die *online*-Kosten direkt von der Performanz des Modells abhängig und die Verwendung aufwändiger Modelle führt schnell zu der Situation, dass die jeweiligen Verfahren nicht praktikabel anwendbar sind. Die Modellgenauigkeit hat direkten Einfluss auf die erreichbare Güte der darauf basierenden modellbasierten Optimierung [91]. Dieser Umstand verschärft sich, wenn Verfahren angewandt werden, die das Modell nutzen um Vorhersagen über einen weiteren Prädiktionshorizont hinweg zu treffen. Selbst ein geringer Modellfehler kann sich dann über die Zeitschritte zu enormen Abweichungen des prädizierten Prozessverhaltens von dem tatsächlichen Prozess führen [36]. Zur Anwendung modellfreier Verfahren ist hingegen kein tieferes Modellwissen erforderlich und Optimierungsergebnisse sind hier nicht durch Annahmen bei der Modellbildung beschränkt. Auf die Unterschiede zwischen modellfreiem bestärkendem Lernen und *modellprädiktiver Regelung* wird in 2.4.1 umfassender eingegangen. Auf der anderen Seite ist ein grundlegender Nachteil modellfreier Verfahren, dass durch das fehlende Modellwissen das Lernproblem erschwert wird, wodurch sie in aller Regel deutlich langsamer konvergieren als modellbasierte Verfahren [36]. Bei der Anwendung auf physikalische Systeme, wie etwa im Bereich der Robotik oder bei der Optimierung von Fertigungsprozessen, entstehen durch das Lernen modellfreier Methoden höhere initiale Kosten im Einsatz, beispielsweise für Energie, durch Verschleiß oder im Fall von Fertigungsprozessen durch Prozessausschuss.

Eine Möglichkeit die Dateneffizienz des modellfreien bestärkenden Lernens deutlich zu verbessern und die Anwendbarkeit auf kontinuierliche Zustandsräume zu ermöglichen, ist die Verwendung approximativer Verfahren (siehe 2.2.7). Sobald sich die Zielstellung des Entscheidungsprozesses ändert, ist jedoch die aktuelle Regelungsstrategie und zugehörige Bewertungsfunktion hinfällig, was für die in Abschnitt 2.2 vorgestellten Algorithmen bedeutet, dass sie von Beginn an eine neue Regelstrategie lernen müssen. Ein Ziel dieser Arbeit ist daher, approximative Methoden zu erforschen und entwickeln, die in Situationen sich ändernder Zielstellungen, oder mehrerer äquivalenter Zielstellungen, Prozesswissen nutzen, das unter abweichenden Zielstellungen generiert wurde, und so neue Regelungsstrategien dateneffizient lernen können.
Diese Methoden basieren, wie auch die in 2.2.7 vorgestellten Methoden, auf dem Persistieren gemachter Erfahrungen in einem *replay-Memory*. Wie in 2.2.4 gezeigt, werden die historischen Erfahrungen des *replay-Memory* durch *off-Policy* Updates zum Lernen verwendet. Wie in 2.2.8 beschrieben, schließt dies *strategiebasierte* Verfahren aus, während bei approximativen *bewertungsbasierten* Verfahren die Nutzung eines *replay-Memory* häufig zentraler Bestandteil ist (siehe 2.2.4). Außerdem existieren einige *off-Policy Actor Critic* Ansätze, die es ermöglichen kontinuierliche Strategien unter Nutzung des *replay-Memories* zu lernen (siehe 2.2.8). *Bewertungsbasierte* Verfahren sind häufig einfacher zu untersuchen als die komplexeren *Actor Critic* Ansätze, da anstelle der beiden gelernten Bestandteile (Bewertungsfunktion und Strategie) nur die gelernten Bewertungsfunktionen beteiligt sind und so insgesamt weniger Hyperparameter vorhanden sind und somit weniger Abhängigkeiten zwischen Hyperparametern bestehen. *Bewertungsbasierte* Verfahren sind jedoch in aller Regel zu *off-Policy Actor Critic* Methoden erweiterbar.
Zusammenfassend ist der Fokus der Arbeit die Entwicklung modellfreier, approximativer, *bewertungsbasierter* Methoden des bestärkenden Lernens zur Optimierung von Regelungsstrategien für Fertigungsprozesse mit endlichem Zeithorizont unter besonderer Berücksichtigung der Dateneffizienz.

Derartige Methoden werden für zwei unterschiedliche Klassen von Fertigungsprozessen entwickelt. In Kapitel 4 werden partiell beobachtbare Fertigungsprozesse mit geringer Episodenlänge betrachtet, wobei die Zielformulierung anhand erwünschter Werkstück-Eigenschaften geschieht. Zur Evaluation werden die entwickelten Methoden zur Optimierung des zeitlichen Verlaufs der Niederhaltekräfte eines Tiefziehprozesses hinsichtlich der gewünschten Werkstück-Eigenschaften instanziiert. Ferner wird das Lernen optimaler Strategien auch unter variierenden Prozessbedingungen (hier exemplarisch der Schmierung) untersucht. Kapitel 5 behandelt die Optimierung von Prozesspfaden, wobei das Ziel anhand vorgegebener Mikrostrukturmerkmale des gewünschten Prozessergebnisses gegeben ist. Der dabei betrachtete allgemeine Deformationsprozess (dargestellt als Sequenz von Deformationsschritten) ist in Hinsicht auf den Zeithorizont, die Kardinalität des Aktionsraums sowie die Dimension der Zustandsbeschreibung von deutlich höherer Komplexität. Tabelle 3.1 fasst die zentralen Unterschiede der zur Untersuchung genutzten exemplarischen Anwendungsfälle aus Kapitel 4 und Kapitel 5 zusammen.

Tabelle 3.1: Gegenüberstellung der Anwendungsfälle der Kernkapitel 4 und 5.

| Charakteristik | Tiefziehen (Kap. 4) | allg. Deformation (Kap. 5) |
|---|---|---|
| Prozessschritte | 5 | bis zu 100 |
| Aktionen | 5 | 201 |
| Observablenraum | $\mathbb{R}^3$ | $\mathbb{R}^{44}$ |
| Partielle Beobachtb. | x | |
| äquivalente Ziele | | x |
| Ziel | Werkstück-Eigensch. | Mikrostruktur |
| Materialmodell | Isotrop | Anisotrop |

Die zentrale Forschungsaufgabe, die in Kapitel 4 behandelt wird ist die Entwicklung und Untersuchung von modellfreien Methoden des bestärkenden Lernens zur *online*-Optimierung von Regelstrategien für partiell beobachtbare Fertigungsprozesse mit geringer Episodenlänge. Hierzu wird aufbauend auf dem Prinzip der Zeitschritt-abhängigen Funktionsapproximationen von ABDP (Kapitel 2.2.4) und einem Lernschema in Anlehnung an NFQ im inkrementellen Modus (Kapitel 2.2.7) ein neuartiger, *bewertungsbasierter* Algorithmus vorgestellt und untersucht. Dieser lernt *online* Regelungsstrategien für einen partiell beobachtbaren Prozess. Neben der methodischen Entwicklung ist eine zentrale technische Herausforderung die Implementierung eines digitalen Prozess-Surrogats, das, basierend auf der Finite-Elemente-Methode simuliert, eine möglichst effiziente und realitätsnahe Experimentalumgebung darstellt.
Der zweite Teil, beschrieben in Kapitel 5, behandelt die Anwendbarkeit von bestärkendem Lernen zur Optimierung von Prozesspfaden zur Erreichung gewünschter Mikrostrukturen eines Werkstück-Materials. In Kombination mit Methoden zur Abbildung von Materialeigenschaften auf Material-Strukturen soll dies die gezielte Entwicklung von Materialien unter der Angabe gewünschter Materialeigenschaften erleichtern. Arbeiten, die ähnliche Fragestellungen behandeln, werden in 2.4.4 aufgeführt. Wie in 2.4.4 erläutert, basieren diese meist auf einer Suche in einer Menge vorberechneter Prozesspfade oder in davon abgeleiteten generalisierenden Strukturen. In einer Veröffentlichung aus dem Jahr 2020 [89] wurde ein Verfahren zur dateneffizienten Optimierung von Prozessparametern vorgeschlagen, das während der Prozesspfad-Optimierung den Prozess abtastet und den sogenannten *active Learning* Algorithmen zuzurechnen ist. Das gleiche Ziel wird auch beim bestärkenden Lernen verfolgt. Dabei wird in [89] das Optimierungsproblem als Suche von zeitunabhängigen Prozessparametern betrachtet und nicht als Entscheidungsprozess formuliert. Durch die Betrachtung als Entscheidungsprozess sind Methoden des bestärkenden Lernens *online* anwendbar und adaptieren Besonderheiten und sich ändernde Bedingungen eines Prozesses. Außerdem erleichtert die Formulierung

als Entscheidungsprozess die Optimierung Zeit-abhängiger Prozessparameter und ist so auch in der Lage sehr lange Prozesspfade zu ermitteln.
Ein zentrales Charakteristikum der Problemklasse ist, dass mehrere äquivalente Ziele (in unserem Fall mehrere Mikrostrukturen, welche die gleichen oder sehr ähnliche Materialeigenschaften aufweisen) existieren. Bei der Methoden-Entwicklung für die Findung optimaler Strategien ist dies zu berücksichtigen. Es gilt also, auf effiziente Art einen Prozesspfad zu ermitteln, der zu einer der äquivalenten Strukturen führt. Wie Tabelle 3.1 entnommen werden kann, sollen gegenüber dem ersten Teil der Arbeit Prozesse mit einer deutlich höheren Anzahl an Prozessschritten berücksichtigt werden. Die Bewertung des Ergebnisses in Form eines Belohnungssignals ungleich Null findet gleichzeitig erst zum Ende einer Prozessausführung statt. Der Umgang mit diesen selten auftretenden Belohnungen stellt eine weitere Herausforderung dieser Problemklasse dar, der im Rahmen der Forschungsaufgabe begegnet werden muss.

# 4 Optimierung partiell beobachtbarer Fertigungsprozesse unter variierenden Einflüssen

In diesem Teil der Arbeit werden Methoden des modellfreien bestärkenden Lernens für die adaptive online Optimierung von Regelungsstrategien partiell beobachtbarer Fertigungsprozesse mit endlichem Zeithorizont unter variierenden Prozessbedingungen behandelt. Die behandelte Problemklasse und Folgerungen für den zugehörigen Markov-Entscheidungsprozess werden in Abschnitt 4.1 formuliert. In Abschnitt 4.2 wird der entwickelte Algorithmus zur effizienten Lösung der spezifizierten Problemklasse eingeführt und behandelt. Zur Untersuchung der entwickelten Methode wird diese für die Optimierung der zeitlich variierten Niederhaltekraft eines Tiefziehprozesses ausgeprägt. Ein interaktives digitales Surrogat des Tiefziehprozesses, basierend auf einer Prozesssimulation, wird in Abschnitt 4.3 vorgestellt. Ergebnisse der Untersuchungen werden in Abschnitt 4.4 vorgestellt. In Abschnitt 4.5 werden Möglichkeiten zur Erweiterung der vorgestellten Methode zur multikriteriellen Optimierung von Fertigungsprozessen besprochen und anhand des Prozess-Surrogats untersucht.
Die hier vorgestellten Methoden und Untersuchungen wurden als Zwischenergebnisse der Arbeit in [80] und [92] veröffentlicht.

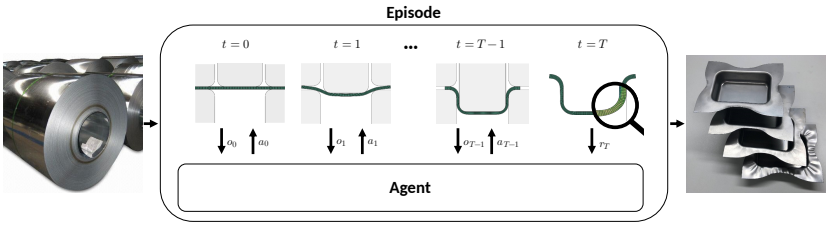



Abbildung 4.1: Schematische Darstellung des Optimierungsproblems anhand des unten betrachteten Anwendungsfalls „Optimierung der Niederhaltekräfte eines Tiefziehprozesses“[1]

## 4.1 Aufgabenstellung

Bei der in diesem Teil der Arbeit behandelten Problemklasse handelt es sich um Fertigungsprozesse mit festem Zeithorizont. Zur Beschreibung als Markov-Entscheidungsprozess wird von einer zeitlich diskreten Prozessregelung ausgegangen. Fertigungsprozesse mit festem Zeithorizont zeichnen sich dadurch aus, dass das Produkt des Prozesses durch eine feste Anzahl $T$ aufeinanderfolgender Prozessschritte mit zugehörigen Regelungsaktionen $a_t$ erzeugt wird. Die Optimierung der Regelungsstrategie zur Erreichung eines Produkts mit gewünschten Eigenschaften für diese Prozesse kann als Markov-Entscheidungsprozess (MDP) mit festem Zeithorizont (siehe 2.2.1) formuliert und durch episodisches bestärkendes Lernen (siehe 2.2.5) gelöst werden. Eine Prozessausführung mit $T$ Regeleingriffen entspricht dabei einer Episode.

Für den Beispielprozess Tiefziehen ist in Abbildung 4.1 der Verlauf einer Episode schematisch dargestellt. Der Agent erhält dabei über $T$ Zeitschritte beobachtbare Größen des Prozesses $o_t$ und führt Regelungsaktionen $a_t$ aus, die den weiteren Prozessverlauf beeinflussen. Am Ende der Episode, in Zeitschritt $t = T$, wird das Prozessresultat bewertet. Den Ergebnissen der Bewertung folgend wird ein Belohnungssignal $r_T$ generiert. Das Belohnungssignal während

[1] Bildquelle Stahlrollen (links in der Abbildung): wikimedia.org, Creative Commons CC BY-SA

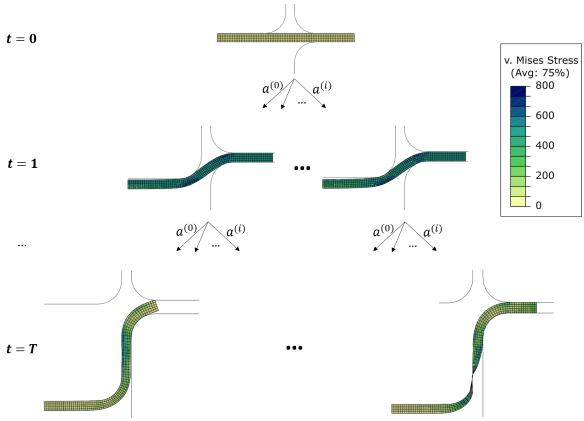



Abbildung 4.2: Graphendarstellung des Markov-Entscheidungsprozess für die deterministische Form des unten betrachteten Anwendungsfalls „Optimierung der Niederhaltekräfte eines Tiefziehprozesses“.

der Prozessausführung ($r_t$ für $t < T$) ist entweder neutral ($r_t = 0$ für $t < T$), oder wird in Abhängigkeit der Aktions-bedingten Prozesskosten gewählt.
Fertigungsprozesse zeichnen sich außerdem in vielen Fällen dadurch aus, dass Prozessaktionen innerhalb des Prozesses irreversibel sind. Beispiele hierfür sind Prozesse, bei denen metallische Werkstoffe plastisch verformt werden und Prozesse der additiven- oder subtraktiven Fertigung. Für diese Prozesse existieren pro Zeitschritt $t \in [0, ..., T]$ zueinander disjunkte Unterräume des Zustandsraums $S_t$. Für einen deterministischen Prozess mit irreversiblen Prozessaktionen und festem Zeithorizont entspricht die Graphendarstellung des zugehörigen Markov-Entscheidungsprozesses einem Baum der Höhe $T$. Dabei entspricht der Ausgangszustand $s_0$ der Wurzel und die Prozessresultate $S_T$ den Blättern des Baums. Wenn außerdem die Menge der ausführbaren Aktionen $A$ zeitunabhängig ist, entspricht es einem Baum mit konstantem Verzweigungsfaktor. Ein solcher Baum ist für den Beispielprozess Tiefziehen in Abbildung 4.2 dargestellt.

In realen Prozessumgebungen kann jedoch selten von deterministischen Prozessen ausgegangen werden. Stattdessen unterscheidet sich jede Prozessausführung durch wechselnde Bedingungen. Solche wechselnden Bedingungen sind beispielsweise gegeben durch das vorliegende Material, die Schmierung des Prozesses und Erwärmung sowie Verschleiß des Werkzeugs. Aus diesem Grund ist davon auszugehen, dass der Prozess variierenden Einflüssen und stochastischen Störgrößen unterliegt. Unter der Annahme, dass sich die Prozessdynamik der Prozessausführungen (Episoden) aufgrund der Einflüsse unterscheidet, während einer einzelnen Prozessausführung (Episode) jedoch ausschließlich von den gewählten Regelungsaktionen abhängt, entspricht die Graphendarstellung des zugehörigen MDPs einem sogenannten *Wald* (einer Menge von Bäumen). Hierbei entspricht eine Zusammenhangskomponente (ein Baum) einer individuellen Anfangsbedingung.
Außerdem zeichnen sich reale Prozessumgebungen dadurch aus, dass der Prozesszustand $s_t$ zum Zeitpunkt $t$ während der Prozessausführung nicht vollständig beobachtbar ist. Stattdessen können von $s_t$ abhängige Größen $\mathbf{o}_t$ gemessen werden, die in der Regel aber nicht ausreichen, um den Zustand $s_t$ zu rekonstruieren. Außerdem ist die Messung der Werte $\mathbf{o}_t$ üblicherweise mit einer Messungenauigkeit versehen. Die formale Beschreibung des Optimierungsproblems als MDP weitet sich dann aus zu einem partiell beobachtbaren MDP (siehe 2.3.1), wobei die Größen $\mathbf{o}_t$ den Zustand $s_t$ nicht vollständig charakterisieren. Aufgrund von Messunsicherheiten liegen einzelne Werte $o_{t,i}$ außerdem nur in verrauschter Form vor, wobei häufig von additivem Gaußschen Rauschen ausgegangen werden kann. $o_{t,i}$ liegt also in der Form

$$o_{t,i} = f(\mathbf{s}_t) + \mathcal{N}(0, \sigma), \tag{4.1}$$

vor. Im Folgenden werden Methoden zur Optimierung der Regelungsstrategie während der Prozessausführung untersucht, die ohne im Voraus gegebenes Prozesswissen modellfrei Lernen. Insbesondere sind die Zustandsübergangsfunktion $P$ und die Beobachtungsfunktion $f : \mathbb{R}^n \to \mathbb{R}$ häufig nicht bekannt.

Aufgabe des Agenten ist dann die Optimierung der Regelungsstrategie anhand der beobachteten Größen $o$ und Belohnungssignale $r$.

## 4.2 Lösungsmethode

In diesem Abschnitt wird ein Algorithmus zur Lösung der im vorangegangenen Abschnitt spezifizierten Aufgabenstellung durch modellfreies, bewertungsbasiertes bestärkendes Lernen vorgestellt: *Backward Fixed-Horizon Neural Q-Learning*[2] (Kurz BFHNQ).

*Backward Fixed-Horizon Neural Q-Learning* (BFHNQ), stellt eine Form des bestärkenden Lernens durch Approximation der Q-Funktion dar. BFHNQ kann als Spezialfall der inkrementell lernenden Variante von *Neural Fitted Q-Approximation* [22] (Listing 6) angesehen werden. Ähnlich dem *Backward Approximate DP* (BADP) Ansatz [2] (Listing 4) werden dabei Approximationen von Zeitschritt-abhängigen Bewertungsfunktionen rückwärts in der Zeit trainiert. Im Gegensatz zu BADP, wo auf diese Art approximierte Zustands-Bewertungsfunktionen $\mathcal{V}_t(s,\theta_t)$ durch Nutzung eines Modells der Zustandsübergangsfunktion gelernt werden, werden durch BFHNQ approximierte Q-Funktionen $\mathcal{Q}_t(\mathbf{s},a,\theta_t)$ gelernt. Hierbei ist kein Modell der Zustandsübergangsfunktion und der Belohnungsfunktion zum Lernen und zur Extraktion von Strategien $\pi$ erforderlich.

Zum Umgang mit der partiellen Beobachtbarkeit von Fertigungsprozessen nutzt BFHNQ den Umstand, dass die Q-Funktion Zeitschritt-abhängig definiert ist. Dabei wird die bisher in der aktuellen Episode, in Form von beobachtbaren Größen $\mathbf{o}$ und ausgeführten Aktionen $a$, ermittelte Information in einer pseudo-Zustandsbeschreibung $\tilde{\mathbf{s}} \in \tilde{S}$ zusammengefasst. Für pseudo-Zustandsbeschreibungen $\tilde{\mathbf{s}}$ wird die *Markov-Annahme* 2.3.1 getroffen, sodass von

[2] Vorgestellt in der Vorveröffentlichung [80] unter dem Namen *Fixed Horizon Manufacturing Process Q-control*

einem MDP mit festem Zeithorizont $(\tilde{S}, A, P, R, \gamma, P_0, T)$ ausgegangen wird. Die Zusammenfassung der bisherigen Observablen und Aktionen geschieht durch Konkatenation der entsprechenden Vektoren. Der Pseudo-Zustand $\tilde{\mathbf{s}}_t$ in Zeitschritt $t = 0$ ist definiert durch $\tilde{\mathbf{s}}_0 = \mathbf{o}_0$ und in allen weiteren Zeitschritten durch

$$\tilde{\mathbf{s}}_t = \tilde{\mathbf{s}}_{t-1} \frown a_{t-1} \frown \mathbf{o}_t, \tag{4.2}$$

wobei der Operator $\frown$ reelle Vektoren konkateniert $\bullet \frown \bullet : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^{m+n}$. Durch die derartige Nutzung der vollständigen Information ist die Dimension des Pseudo-Zustandsraums abhängig von $t$. Für $a \in \mathbb{R}$ und $\mathbf{o} \in \mathbb{R}^n$ gilt $\tilde{\mathbf{s}}_t \in \mathbb{R}^{t+(t+1)n}$. Für die vorgeschlagene Form der pseudo-Zustände wäre eine herkömmliche, Zeitschritt-übergreifende Approximation der Q-Funktion $\mathcal{Q}(\tilde{\mathbf{s}}, a, \theta)$ aus diesem Grund nicht geeignet. Eine separate Approximation $\mathcal{Q}_t(\tilde{\mathbf{s}}_t, a, \theta_t)$ pro Zeitschritt ist aber ohne Weiteres mit der vorgeschlagenen Form vereinbar.

Der gesamte BFHNQ Algorithmus ist in Listing 8 aufgeführt. Obwohl der Fokus in diesem Teil der Arbeit auf Entscheidungsprozessen mit festem Zeithorizont liegt, ist BFHNQ durch kleine Änderungen im Kontrollfluss auf Markov-Entscheidungsprozesse mit endlichem Zeithorizont erweiterbar. Die entsprechenden Änderungen sind im Listing in roter Farbe markiert. Sie betreffen den Kontrollfluss im Fall, dass die Binär-Variable $d$ angibt, dass die aktuelle Episode beendet ist (Zeile 12). Der Erwartungswert für den aktuellen pseudo-Zustand wird dann als 0 definiert (Zeile 13). Anschließend wird die aktuelle Episode beendet (Zeile 14). Im weiteren Verlauf des Kapitels, sowie in der Behandlung des Anwendungsfalls spielt dieser Fall keine weitere Rolle und es wird von Entscheidungsprozessen mit festem Zeithorizont ausgegangen.

Initial wird die Ausgabe der Q-Funktion $\mathcal{Q}_T$ zum finalen Zeitschritt $T$ als Konstant 0 definiert (Zeile 1). Die weiteren approximierten Q-Funktionen $\mathcal{Q}_t$ können durch Setzen der Parameter $\theta_t$ beliebig initialisiert werden (Zeile 3). Nach der Initialisierung der Q-Funktionen $\mathcal{Q}_t$ und der Zeitschritt-abhängigen *Replay*

BACKWARDFHNQ$(\varepsilon_0, \lambda_\varepsilon, \alpha, n_e, n_\mathcal{Q})$

```
1   Q_T(s_T, a, θ_T) := 0,  ∀s_T ∈ S_T, ∀a ∈ A, ∀θ_T
2   for t = 0 to T − 1
3       initialize Q_t parameters θ_t
4       D_t ← ∅
5   for e = 1 to n_e
6       observe initial observable o
7       s̃ ← o
8       for t = 0 to T − 1
9           execute a following π̃ and observe (o′, r, d)
10          s̃′ ← s̃ ⌢ a ⌢ o′
11          D_t ← D_t ∪ {(s̃, a, s̃′)}
12          if d = 1:
13              Q_{t+1}(s̃′, a, θ_t) := 0,  ∀a ∈ A, ∀θ_t
14              go to line 16
15          s̃ ← s̃′
16      if e divides n_Q:
17          for t = T − 1 to 0
18              Learn parameters θ_t of Q_t based on D_t and Q_{t+1}.
```

Listing 8. *Backward Fixed-Horizon Neural Q-Learning*. Rot markierte Code-Bestandteile: Ergänzung zum *Backward Finite-Horizon Neural Q-Learning*

*Memories* $\mathcal{D}_t$ (Zeilen 1 bis 4) wird für $n_e$ Episoden der Lernalgorithmus ausgeführt (Zeilen 5 bis 18). Dabei werden die pseudo-Zustände (4.2) folgend gebildet (Zeilen 7, 10, 15). Die Interaktion mit dem Prozess (Zeilen 6, 9) folgt dem Interaktions-Zyklus des bestärkenden Lernens (siehe 2.2.5). Die Lernstrategie $\tilde{\pi}$ ist $\varepsilon$-*greedy* (siehe (2.14) in 2.2.6) bezüglich den approximierten Q-Funktionen $\mathcal{Q}_t$.

Nach jeweils $n_\mathcal{Q}$ Episoden (Zeile 16) werden die Approximationen der Zeitschritt-abhängigen Q-Funktionen neu gelernt. Das Neu-Lernen wird dabei wie oben besprochen rückwärts in den Zeitschritten durchgeführt (Zeile 17). Das Netz $\mathcal{Q}_t$ wird basierend auf dem *Replay Memory* $\mathcal{D}_t$ und der bereits trainierten Approximation $\mathcal{Q}_{t+1}$ trainiert (wobei für $t = T-1$ gilt $\mathcal{Q}_{t+1} = \mathcal{Q}_T = 0$). Das

Trainings-Loss pro Erfahrungs-Tupel $(\tilde{\mathbf{s}}_t, a_t, \tilde{\mathbf{s}}_{t+1}, r, d)$ ist die quadrierte Abweichung $\big(\mathcal{Q}_t(\tilde{\mathbf{s}}_t, a_t, \theta_t) - y_\mathcal{Q}\big)^2$ der bisherigen Q-Wert-Schätzung von dem Zielwert[3]

$$y_\mathcal{Q} = \mathcal{Q}_t + \alpha\big(r + \gamma \max_{a_{t+1} \in A} \mathcal{Q}_{t+1}(\tilde{\mathbf{s}}_{t+1}, a_{t+1}, \theta_{t+1}) - \mathcal{Q}_t\big). \tag{4.3}$$

## 4.3 Prozessinstantiierung

Der im Vorangegangenen vorgestellte BFHNQ Algorithmus wird zur Untersuchung und Evaluation für die Optimierung eines Tiefziehprozesses ausgeprägt. Der Erfolg der Fertigung und die Eigenschaften des Resultats eines Tiefziehvorgangs hängen wesentlich von der Kraft ab, die der Niederhalter während des Vorgangs auf das Werkstück ausübt. Aus diesem Grund ist die Wahl zeitlich und/oder räumlich variierender Niederhaltekräfte Gegenstand einiger verwandter Forschungsarbeiten zur Optimierung und der optimalen Regelung des Tiefziehprozesses (siehe 2.4.3).

Der Aufgabenstellung aus Abschnitt 4.1 folgend, wird die modellfreie Optimierung zeitlich variierender Niederhaltekräfte durch BFHNQ untersucht. Eine Episode entspricht dabei, wie in Abbildung 4.1 bereits skizziert, einem Tiefziehvorgang mit anschließender Begutachtung des Tiefzieh-Ergebnisses. Grundlage der Untersuchungen ist ein Finite-Elemente-Simulationsmodell des Tiefziehprozesses, das in 4.3.1 beschrieben wird. Fokus der Untersuchungen ist die modellfreie Optimierung der Regelstrategie während der Prozessausführung. Wie beschrieben zeichnet sich ein solches Optimierungsproblem insbesondere durch stochastisches Prozessverhalten und eine eingeschränkte Beobachtbarkeit des Prozesszustands aus. Um diese Eigenschaften einer Realumgebung nachzustellen, werden für die Untersuchungen die Simulationsläufe

[3] Bei der Anwendung von BFHNQ herausgestellt, dass eine Lernrate $\alpha$ kleiner 1 der Stabilität des Lernens zuträglich sein kann.

mit stochastisch gewählten Prozesseinflüssen initialisiert und aus den Ergebnissen der Simulation beobachtbare Größen abgeleitet. Die Modellierung der Prozesseinflüsse wird in 4.3.2 beschrieben. Observable Größen werden in 4.3.3 spezifiziert. Die Ergebnisse der Begutachtung eines tiefgezogenen Werkstücks wird durch die in 4.3.4 vorgestellte Belohnungsfunktion quantifiziert. Die Implementierung der Untersuchungsumgebung wird in 4.3.5 erläutert. Bei den Untersuchungen verwendete Parameter des BFHNQ Algorithmus und Hyperparmeter der verwendeten $\mathcal{Q}_t$-Netze sind in 4.3.6 aufgeführt.

### 4.3.1 Simulationsmodell

Zur Simulation des Tiefziehprozesses wird unter der Annahme der Rotationssymmetrie des Prozesses ein 2-dimensionales Finite-Elemente-Modell (FE-Modell) verwendet[4]. Unter der Annahme, dass das Material isotrope Verformungseigenschaften aufweist, bildet das verwendete elastisch-plastische Materialmodell die Eigenschaften des $Fe$-28$Mn$-9$Al$-0.8$C$ Stahls, folgend [93], ab. Die getroffenen Annahmen ermöglichen die Simulation des Tiefziehprozesses in verhältnismäßig kurzer Zeit und somit eine umfassende Untersuchung der Lernalgorithmen.
Das FE-Modell umfasst drei Werkzeuge und ein Bauteil und ist, zusammen mit den unten angeführten beobachtbaren Größen $o$ und der Aktionsgröße $a$, in Abbildung 4.3 dargestellt. Beim Tiefziehen wird ein Blechzuschnitt (graues FE-Netz) durch einen Stempel (blaues Werkzeug) in eine Matrize (grünes Werkzeug) gedrückt. Der Niederhalter (rotes Werkzeug) drückt das Blech auf die Matrize. Das Werkstück hat eine Stärke von 2.5$mm$ und einen Radius von 40$mm$. Der Stempel hat einen Radius von 20$mm$. Der Vorschub des Stempels

[4] Das verwendete FE-Modell stammt aus der Veranstaltung Einführung in die FEM des Institut für Technische Mechanik am Karlsruher Institut für Technologie (KIT)

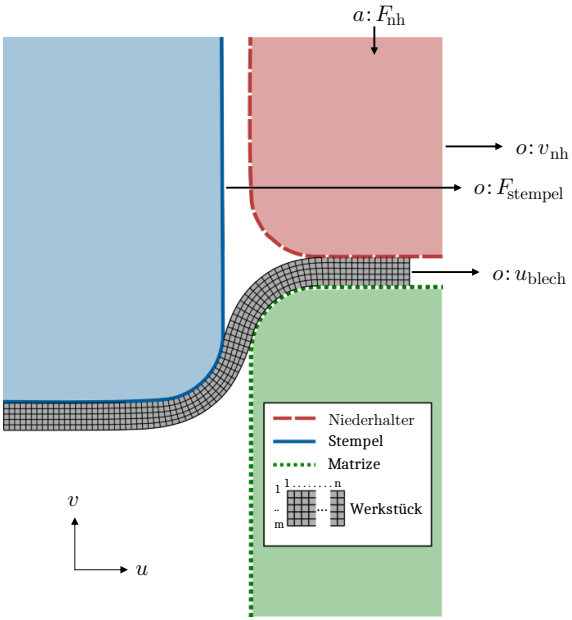



Abbildung 4.3: Darstellung der Werkzeuge und des Bauteils des verwendeten Simulationsmodells, der beobachtbaren Größen $o$ und der Eingangsgröße $a$

beträgt 25*mm* und wird mit konstanter Geschwindigkeit durchgeführt. Das deformierbare Werkstück ist in 5 mal 80 rotationssymmetrische Vierknoten-Kontinuumselemente mit reduzierter Integration (Abaqus CAX4R) aufgeteilt. Die Werkzeuge sind als Starrkörper modelliert.

Die Niederhaltekraft (*Blank Holder Force*, BHF) $F_{\text{nh}}$ ist zu sechs äquidistanten Zeitpunkten definierbar. Werte zwischen den diskreten Zeitpunkten sind linear interpoliert. Die Zeitpunkte entsprechen dem Beginn einer Episode $t = 0$ und $T = 5$ aufeinanderfolgenden Prozessschritten $t \in [1, 2, ..., 5]$. Die Aktion $a_t$ in Zeitschritt $t$ legt die Niederhaltekraft im nächsten Zeitschritt fest. Zur

Anwendung des BFHNQ Algorithmus wurden die dabei wählbaren Werte diskretisiert und betragen $[20kN, 40kN, ..., 140kN]$. Die initiale Niederhaltekraft beträgt $0kN$

### 4.3.2 Stochastische Störgrößen

Das Prozessverhalten beim Tiefziehen variiert unter anderem aufgrund der Schmierung und Reibung an den Kontaktstellen der beteiligten Werkzeuge mit dem Werkstück [77]. Bei den Untersuchungen wird dies berücksichtigt, indem der Reibungskoeffizient $\mu$ als stochastische Größe modelliert wird. Der Reibungskoeffizient $\mu$ wird pro Episode $e$ aus einer skalierten und diskretisierten Beta-Verteilung zufällig gezogen. Diese ist für $0 \geq x \geq 1$ durch die Wahrscheinlichkeitsdichte

$$f(x) = \frac{1}{B(p_\beta, q_\beta)} x^{p_\beta - 1}(1-x)^{q_\beta - 1} \tag{4.4}$$

definiert, wobei die Beta-Funktion $B$ eine Normalisierungskonstante darstellt und die Parameter $p_\beta \in \mathbb{R}, q_\beta \in \mathbb{R}$ die Ausprägung der Beta-Verteilung bestimmen.
Die für den Reibungskoeffizienten angenommene Beta-Verteilung, ist definiert durch die Parameter $p_\beta = 1.75$ und $q_\beta = 5$ und so skaliert, dass sie den Wertebereich $[0, 0.14]$ annimmt. Zur mehrfachen Verwertung der Simulationsergebnisse (siehe 4.3.5) während der Untersuchungen wurde die Verteilung außerdem diskretisiert, so dass die resultierende diskrete Verteilung auf einer Menge von 10 äquidistanten Werten im Intervall $[0.014, 0.14]$ definiert ist. Die resultierende Wahrscheinlichkeitsfunktion weist einen Modus von $0.028$ auf und ist in Abbildung 4.4 dargestellt.

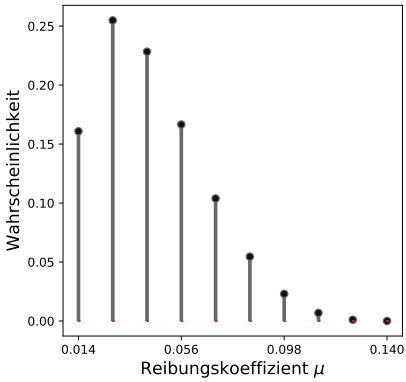



Abbildung 4.4: Darstellung der diskretisierten Beta-Verteilung des Reibungskoeffizienten $\mu$.

### 4.3.3 Observable Größen

Der Zustand $s_t$ für $t > 0$ ist abhängig von $s_{t-1}$, der Aktion $a_{t-1}$ und dem Reibungskoeffizienten $\mu$. Bis auf $a_{t-1}$ sind diese Werte dem Agenten nicht bekannt. Stattdessen ist in jedem Zeitschritt ein Vektor $\mathbf{o}_t = (F_{\text{stempel}}, u_{\text{blech}}, v_{\text{nh}})^\intercal$, bestehend aus drei beobachtbaren Größen gegeben:

- der aktuellen Stempelkraft $F_{\text{stempel}}$,
- dem aktuellen Einzug des Bleches $u_{\text{blech}}$,
- der aktuellen Position des Niederhalters in $v$-Richtung $v_{\text{nh}}$.

Die Messunsicherheit ist durch additives Gaußsches Rauschen modelliert. Die hierfür gewählte Standardabweichung beträgt 1% der jeweils experimentell ermittelten empirischen Spanne von $F_{\text{stempel}}$ und $u_{\text{blech}}$, sowie 0.5% der experimentell ermittelten empirischen Spanne von $v_{\text{nh}}$.

### 4.3.4 Belohnungsfunktion

Wie für die Problemklasse in Abschnitt 4.1 beschrieben, wird nach Abschluss der Prozessausführung das Resultat begutachtet und die Ergebnisse der Begutachtung durch die Belohnungsfunktion quantifiziert. Für den beschriebenen Tiefziehprozess setzt sich das Belohnungssignal zum Zeitpunkt $T$ aus drei Qualitätskriterien zusammen:

- Den Eigenspannungen des tiefgezogenen Werkstücks
- Der minimalen Wandstärke des tiefgezogenen Werkstücks, und
- Dem Materialverbrauch (Einzug).

Der Zustand $\mathbf{s}_T$ zur Berechnung des Belohnungssignals $r_T = R(\mathbf{s}_T)$ setzt sich aus drei Matrizen $\mathbf{M}, \mathbf{H}, \mathbf{D} \in \mathbb{R}^{m \times n}$ zusammen, die jeweils Werte der $m = 5$ und $n = 80$ Flächenelemente des Bleches beinhalten. Der Wert $M_{ij}$ gibt dabei die mittlere *Von-Mises-Vergleichsspannung* des Flächenelements $(ij)$, $H_{ij}$ die Höhe bezüglich der Ausgangslage des Flächenelements $(ij)$ und $D_{ij}$ die Verschiebung in $u$-Richtung bezüglich der Ausgangslage des Flächenelements an Stelle $(ij)$ an.
Basierend auf der Zustandsbeschreibung sind drei Kostenterme $c_{\text{mises}}$, $c_{\text{wand}}$, $c_{\text{verbrauch}} \in \mathbb{R}$ zur Quantifizierung der oben beschriebenen Qualitätskriterien definiert. Der Kostenterm $c_{\text{mises}}$ bewertet die Materialspannungen und ist definiert als

$$c_{\text{mises}} = \|\mathbf{M}\|_F \,, \tag{4.5}$$

wobei $\|\mathbf{M}\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{M}_{ij}{}^2}$ der Frobeniusnorm der Matrix $\mathbf{M}$ entspricht. Der Kostenterm $c_{\text{wand}}$ gibt die minimale Wandstärke an

$$c_{\text{wand}} = -\min(\mathbb{1}_n \mathbf{H}), \tag{4.6}$$

wobei $\mathbb{1}_n \in \mathbb{R}^n$ dem Einsvektor entspricht.

Der Kostenterm $c_{\text{verbrauch}}$ reflektiert den Materialverbrauch als Summe der Verschiebungen der äußeren Elemente des Bleches

$$c_{\text{verbrauch}} = \sum_{i=1}^{m} \mathbf{D}_{in}. \tag{4.7}$$

Zur Erzeugung des skalaren Belohnungssignals werden zu den Kosten $c_l, l \in \text{mises}, \text{wand}, \text{verbrauch}$ Belohnungsterme $\mathfrak{r} = [\mathfrak{r}_{\text{mises}}, \mathfrak{r}_{\text{wand}}, \mathfrak{r}_{\text{verbrauch}}]^\intercal$ berechnet. Die Belohnungsterme sind gegeben als

$$\mathfrak{r}_l = 1 - \frac{c_l - c_l^{\min}}{c_l^{\max} - c_l^{\min}}, \tag{4.8}$$

wobei $c_l^{\min}, c_l^{\max} \in \mathbb{R}$ jeweils empirisch ermittelte Minimal- und Maximalwerte des Kostenterms $c_l$ sind.
Zur Berechnung eines skalaren Belohnungssignals wird das gewichtete harmonische Mittel

$$H(\mathbf{x}, \mathbf{w}) = \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} \frac{w_i}{x_i}} \tag{4.9}$$

auf die Belohnungsterme $\mathfrak{r}$ angewandt. Das harmonische Mittel wird verwendet, um Prozessergebnisse mit ausgeglichenen Eigenschaften bezüglich der Belohnungsterme zu bevorzugen. Der Gewichtsvektor $\mathbf{w}$ mit $\|\mathbf{w}\|_1 = 1$ kann dabei genutzt werden, um den Einfluss der einzelnen Kostenterme in der Anwendung zu bestimmen.
Für $t < T$ und im Fall, dass ein Belohnungswert $\mathfrak{r}_i$ kleiner 0 vorliegt gilt $R(\mathbf{s}_t) = 0$. Die Belohnungsfunktion ist damit gegeben als

$$R(\mathbf{s}_t) = \begin{cases} 10 \times H(\mathfrak{r}, \mathbf{w}) & \text{, if } t = T \wedge \min(\mathfrak{r}) > 0, \\ 0 & \text{, else.} \end{cases} \tag{4.10}$$

Die Skalierung der Belohnung um den Faktor 10 ist willkürlich gewählt und erleichtert die Approximation der Bewertungsfunktionen durch künstliche Neuronale Netze. In Abschnitt 4.5 werden auf dieser Basis Untersuchungen zur multikriteriellen Optimierung vorgestellt. Wenn nicht abweichend angegeben, wird für die vorgestellten Untersuchungen von einer Gleichgewichtung der drei Terme ausgegangen.
In Abbildung 4.5 ist die Höhe der Werte der drei Belohnungsterme in Abhängigkeit von der Abfolge der Niederhaltekräfte, im Folgenden auch als Prozesspfad bezeichnet, dargestellt. Prozesspfade wurden mit dem in 4.3.1 definierten Modell mit einem deterministischen Reibungskoeffizienten von $\mu = 0.028$ (entspricht dem Modus der in 4.3.2 beschriebenen Verteilung) simuliert. Der Wert des jeweiligen Belohnungsterms ist durch den Farbwert des Prozesspfads kodiert. Die Prozesspfade mit den drei höchsten Belohnungsterm-Werten sind jeweils hervorgehoben dargestellt.

### 4.3.5 Implementierung

Zur Untersuchung der in dieser Arbeit entwickelten Methoden anhand simulierter Fertigungsprozesse wurde eine generische Experimentalumgebung entwickelt, die die effiziente, flexible und reproduzierbare Ausführung von Experimenten ermöglicht. Die Schichten und Module der generischen Umgebung sowie ein schematischer Informationsfluss zwischen den Schichten sind in Abbildung 4.6 dargestellt. Die generische Umgebung implementiert die *OpenAI Gym* Schnittstelle [94], wodurch insbesondere die Kommunikation mit dem Agenten in standardisierter Form geschieht. Diese Kommunikation geschieht aus Sicht des Agenten im allgemeinen Fall insbesondere durch das initiale Zurücksetzen der Umgebung und die Beobachtung der initialen beobachtbaren Werte (siehe Listing 8 Zeile 6) und während der Episode durch das Ausführen von Aktionen und die Beobachtung des Belohnungssignals $r$, der beobachtbaren Werte $\mathbf{o}'$ und des Indikators $d$ (siehe Listing 8 Zeile 9).

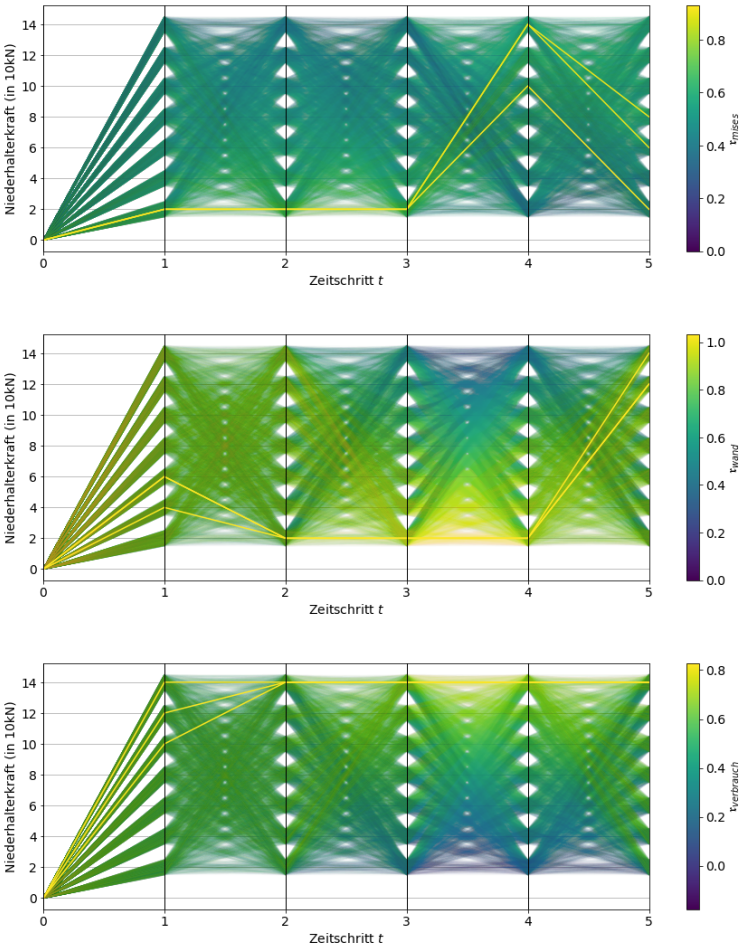



Abbildung 4.5: Wert der Belohnungsterme $\mathfrak{r}_{\text{mises}}$, $\mathfrak{r}_{\text{wand}}$, $\mathfrak{r}_{\text{verbrauch}}$ (in der Reihenfolge der Darstellung) für alle $7^5$ möglichen Prozesspfade mit einem Reibungskoeffizienten von $\mu = 0.028$. Pro Belohnungsterm ist jede Abfolge der Niederhaltekräfte dargestellt, wobei die Höhe des jeweiligen Terms durch die Farbe repräsentiert ist. Die drei Abfolgen mit den höchsten Werten des jeweiligen Belohnungsterms sind mit einer erhöhten Strichstärke dargestellt, alle weiteren halbtransparent mit einem Alphawert von 0.01. Prozesspfade sind absteigend nach Höhe der Werte des jeweiligen Belohnungsterms sortiert dargestellt. Die diskreten Werte auf der $y$-Achse (Niederhaltekraft, $[20kN, 40kN, ..., 140kN]$) wurden ab dem Zeitschritt $t = 1$ zu Gunsten der Darstellung manuell verrauscht.

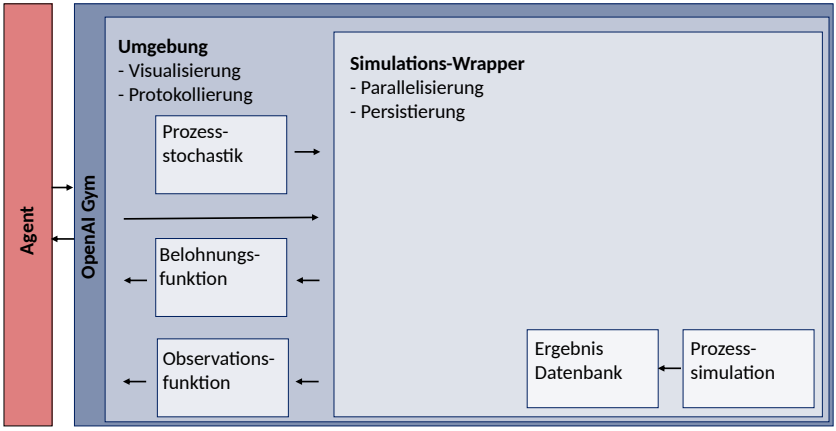



Abbildung 4.6: Architektur und schematischer Informationsfluss der simulationsbasierten Umgebung

Abbildung 4.6 stellt eine Draufsicht der Software-Schichten dar, wobei die Helligkeit der Farbe die Höhe der jeweiligen Schicht repräsentiert. Der Informationsfluss findet dabei hierarchisch von unten (*OpenAI Gym*) nach oben (*Prozesssimulation*) und umgekehrt statt. Die Informationsflüsse zwischen Modulen auf gleicher Ebene sind durch Pfeile gekennzeichnet. Die *Umgebung* implementiert die *OpenAI Gym* Schnittstelle und kümmert sich unter anderem um die Protokollierung und Visualisierung des Experiment-Fortschritts, sowie die Vermittlung zwischen Agent und simuliertem Prozess. Aktionen des Agenten werden zusammen mit zufällig erzeugten Prozessbedingungen an den *Simulations-Wrapper* weiter gereicht.

Der *Simulations-Wrapper* vermittelt zwischen Umgebung und Prozesssimulation und kümmert sich um die *Persistierung* der Simulations-Ergebnisse und -Zwischenergebnisse, sowie die konfliktfreie parallele Durchführung von Simulationen. Die notwendigen Simulationsrechnungen, beispielsweise durch die Finite-Elemente-Methode, sind häufig sehr rechenintensiv. Der simulierte Zustand in jedem Zeitschritt einer Episode ist abhängig von bisher ausgeführten

Aktionen $a$ und den stochastischen Störgrößen und kann in Form des Simulationszustandes persistiert werden. Wenn, wie im hier vorliegenden Fall, sowohl Aktionen (siehe Kapitel 4.3.1) als auch Prozessbedingungen (siehe 4.3.2) Elemente einer diskreten Menge sind, können Simulationszustände einer Episode bei weiteren Episoden mit identischen Prozessbedingungen ausgehend von $t = 0$ bis zu einem Punkt, an dem die Episoden sich unterscheiden, wiederverwendet werden. Aus diesem Grund werden Simulationszustände in jedem Zeitschritt der Episode persistiert. Eine weitere Strategie neben der *Persistierung* ist die *Parallelisierung* der Simulationen. So ist es beispielsweise möglich, mehrere Experimente parallel durchzuführen und so gegebene Rechenkapazitäten effizient zu nutzen. Der *Simulations-Wrapper* kümmert sich dabei um die Vermeidung von Konflikten bei der Durchführung der Simulation und dem Zugriff auf Simulationsergebnisse. Nachdem eine Aktion $a$ durch den *Simulations-Wrapper* behandelt wurde, werden die Belohnungsfunktion und Observations-Funktion auf die Simulationsergebnisse angewandt und die resultierenden Größen $(r, \mathbf{o}', d)$ zurück-gegeben.
Für die in diesem Kapitel beschriebenen Untersuchungen wird die Tiefziehsimulation mittels ABAQUS durchgeführt, wobei für die Interaktion während der Episode die ABAQUS *Python*-Schnittstelle (siehe [95]) in Kombination mit der *restart*-Funktionalität (siehe [96], Kapitel 9.1.1) genutzt wird. Die Simulationszeit beträgt dabei für zwei 2.6 GHz Prozessorkerne in etwa 60 Sekunden.

### 4.3.6 Versuchsaufbau, Netzarchitekturen und Parameter

Zur Untersuchung des BFHNQ Algorithmus wurde dieser auf den im Vorangegangenen beschriebenen, simulierten Tiefziehprozess angewandt. Ergebnisse der Untersuchungen werden in 4.4.1 vorgestellt. Wenn nicht explizit abweichend angegeben wurden bei den Untersuchungen die folgenden BFHNQ-Parameter verwendet:

- BFHNQ wird mit einer Lernrate von $\alpha = 0.7$ ausgeführt.
- Die Explorationsrate der Lernstrategie in Episode $e$ ist definiert entspricht $\varepsilon = \varepsilon_0 \times \exp(-\lambda_\varepsilon e)$. für $\varepsilon_0 = 0.3$ und die Zerfallsrate $\lambda_\varepsilon = 10^{-3}$. Sie ist definiert durch die initiale Explorationsrate $\varepsilon_0 = 0.3$ und die Zerfallsrate $\lambda_\varepsilon = 10^{-3}$.
- Der Diskontierungsfaktor beträgt $\gamma = 1$.
- Die $\mathcal{Q}_t$-Netze werden nach jeweils $n_\mathcal{Q} = 50$ Episoden trainiert.

Die künstlichen Neuronalen Netze zur Approximation der Q-Funktion ($\mathcal{Q}_t$-Netze) sind *feedforward* Netze und werden Batch-weise mittels L-BFGS trainiert (siehe Anhang A.0.2). Hyperparameter und Besonderheiten beim Training sind im Einzelnen:

- Die $\mathcal{Q}_t$-Netze verfügen über jeweils zwei versteckte Schichten mit ReLU Aktivierungsfunktion. Die Anzahl der Neuronen beträgt 10 in jeder versteckten Schicht für $\mathcal{Q}_1$ und 50 in jeder versteckten Schicht für $\mathcal{Q}_2, \mathcal{Q}_3, \mathcal{Q}_4$.
- Zum Lernen der $\mathcal{Q}_t$-Netze werden die Belohnungen quadriert, da sich während der Untersuchungen herausgestellt hat, dass dies zu verbesserter Dateneffizienz führt. In 4.4.1 dargestellte Ergebnisse und Abbildungen basieren auf den nicht-quadrierten Belohnungssignalen der in 4.3.4 beschriebenen Belohnungsfunktion.
- Bei dem Training der $\mathcal{Q}_t$-Netze wird $L2$-Regularisierung angewandt.

Die Q-Funktion für den Zeitpunkt $t = 0$, $\mathcal{Q}_0$ stellt einen Spezialfall dar und wird nicht durch ein Neuronales Netz approximiert. Da bei dem beschriebenen Anwendungsfall die beobachtbaren Größen zu Beginn der Episode keine Information bezüglich der Prozessbedingungen beinhalten, unterscheidet sich die Beschreibung des Startzustands $\mathbf{s_0}$ nicht zwischen den Episoden. Die Bewertungsfunktion $\mathcal{Q}_0$ wird deshalb als tabellarische Q-Funktion über die Aktionen $a \in A$ gelernt.

Die Parameter und Hyperparameter zur Untersuchung der Dateneffizienz im deterministischen Fall, vorgestellt in 4.4.2, weichen in den folgenden Punkten ab:

- BFHNQ wird mit einer Lernrate von $\alpha = 1$ und einer statischen Explorationsrate von $\varepsilon = 0.1$ ausgeführt.
- Die Kapazität der $\mathcal{Q}_t$-Netze ist aufgrund der vereinfachten Aufgabenstellung reduziert. $\mathcal{Q}_1$ und $\mathcal{Q}_2$ bestehen aus jeweils einer versteckten Schicht mit 5 Neuronen bei $\mathcal{Q}_1$ und 10 Neuronen bei $\mathcal{Q}_2$. $\mathcal{Q}_3$ und $\mathcal{Q}_4$ bestehen aus jeweils zwei versteckten Schichten mit 10 Neuronen im Fall von $\mathcal{Q}_3$ und 50 Neuronen bei $\mathcal{Q}_4$.
- Die $\mathcal{Q}_t$-Netze werden nach jeweils $n_{\mathcal{Q}} = 10$ Episoden trainiert.
- Der Reibungskoeffizient beträgt konstant 0.028.

## 4.4 Ergebnisse

### 4.4.1 Untersuchung des stochastischen Falls mit partieller Beobachtbarkeit

Im Folgenden werden Untersuchungsergebnisse vorgestellt, die durch Anwendung des in Abschnitt 4.2 vorgestellten BFHNQ Algorithmus auf den in Abschnitt 4.3 beschriebenen Tiefziehprozess erzeugt wurden. Während der Untersuchung verwendete Parameter des Algorithmus und Hyperparameter der $\mathcal{Q}_t$-Netze sind in 4.3.6 spezifiziert. Aufgrund der Stochastik der Lernstrategie $\tilde{\pi}$, der Prozessbedingungen (siehe 4.3.2) und der Prozessbeobachtung (siehe 4.3.3) wurden Optimierungsläufe im Rahmen der Untersuchungen zur zuverlässigen Quantifizierung der Ergebnisse wiederholt durchgeführt. Experimente bestehen aus jeweils zehn unabhängigen Optimierungsläufen. Als Baseline-Verfahren dient eine hypothetische nicht-adaptive Methode zur Bestimmung

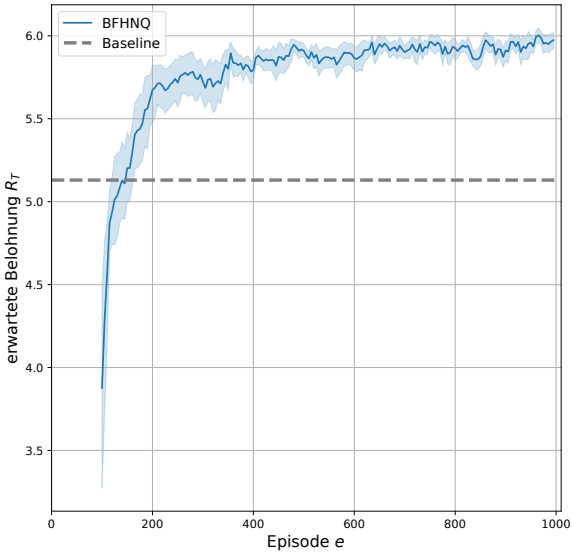



Abbildung 4.7: Mittelwert und 95%-Konfidenzintervall der erwarteten Belohnung $R_T$ in Abhängigkeit von der Episode $e$ für zehn unabhängige BFHNQ Optimierungsläufe und die modellbasierte *Baseline*. Die erwartete Belohnung entspricht der mittleren Belohnung einer nicht-explorativen Strategie, die *greedy* den in $e$ vorliegenden $\mathcal{Q}_t$-Netzen folgt.

der optimalen Niederhaltekräfte im Vorfeld der Prozessausführung. Für diese wird angenommen, dass sie über ein exaktes und wahres Prozessmodell verfügt, das allerdings von einem deterministischen Prozess ausgeht, so dass die Prozesseinflüsse durch das Baseline-Verfahren nicht berücksichtigt werden. Der angenommene deterministische Prozess besitzt einen Reibungskoeffizienten von 0.028. Dies entspricht dem Modus der diskretisierten Verteilung des Reibungskoeffizienten für den simulierten Tiefziehprozess (siehe Abbildung 4.4). Das nicht-adaptive Baseline-Verfahren führt unter dieser Annahme eine Abfolge von Niederhaltekräften aus, die bezüglich des deterministischen Prozessmodells optimal ist. Diese Abfolge wurde durch eine vollständige Suche im Lösungsraum ermittelt. Die Baseline-Belohnung beträgt 5.13 und entspricht

dem Erwartungswert der Belohnung pro Episode, wenn die ermittelte Abfolge für den Surrogat-Prozess mit stochastischem Reibungskoeffizienten ausgeführt wird.

Die Ergebnisse der Anwendung des BFHNQ Algorithmus in 10 unabhängigen Optimierungsläufen auf das partiell beobachtbare Surrogat des Tiefziehprozesses mit stochastischen Einflüssen sind in Abbildung 4.7 dargestellt. Jeder Optimierungslauf besteht dabei aus $n_e = 1000$ Episoden. Ergebnisse sind in Form des Mittelwerts und des 95%-Konfidenzintervalls der erwarteten Belohnung pro Episode angegeben. Die erwartete Belohnung in Episode $e$ entspricht dem Erwartungswert der Belohnung für eine nicht-explorative (*greedy*) Strategie, die den jeweils in Episode $e$ vorliegenden $\mathcal{Q}_t$-Netzen folgt, in Abhängigkeit von der Verteilung des Reibungskoeffizienten. Bei der Ermittlung der Werte wurden, pro Reibungskoeffizient $\mu$, die jeweils letzten *greedy*-Ausführungen aus den Episoden $0, ..., e$ berücksichtigt. Die graue gestrichelte Linie entspricht der erwarteten Belohnung des oben beschriebenen Baseline-Verfahrens.

Um den Effekt der BFHNQ-Parameter auf das Lernverhalten zu untersuchen, wurden Experimente mit variierender Lernrate $\alpha$ und variierender initialer Explorationsrate $\varepsilon_0$ durchgeführt. Pro Parameter-Kombination wurden 10 unabhängige Optimierungsläufe, jeweils bestehend aus 2500 Episoden, mit den entsprechend parametrisierten Algorithmus ausgeführt. Die Auswirkungen der Parameter haben sich dabei wie folgt dargestellt.

Eine steigende Explorationsrate hat einen leicht negativen Einfluss auf die erhaltene Belohnung pro Episode. Der mittlere beobachtete Ertrag pro Episode $\hat{r}_\mu$ während der ersten 250 Episoden beträgt $\hat{r}_\mu = 4.69$ für $\varepsilon_0 = 0.4$ und $\hat{r}_\mu = 5.03$ für $\varepsilon_0 = 0.1$. In späteren Episoden nähern sich die Werte zwar an, das erhöhte Explorationsverhalten für $\varepsilon_0 = 0.4$ führt allerdings nicht zu einem sichtbaren Effekt bezüglich der Qualität der ermittelten Strategie. Die mittlere beobachtete Belohnung pro Episode in den Episoden $e = 1250$ bis $e = 1500$ liegt bei $\hat{r}_\mu = 5.75$ für $\varepsilon_0 = 0.4$ und bei $\hat{r}_\mu = 5.86$ für $\varepsilon_0 = 0.1$ und während der letzten 250 Episoden bei $\hat{r}_\mu = 5.89$ für $\varepsilon_0 = 0.4$ und $\hat{r}_\mu = 5.95$ für $\varepsilon_0 = 0.1$. Dies lässt die Folgerung zu, dass für den betrachteten Anwendungsfall BFHNQ

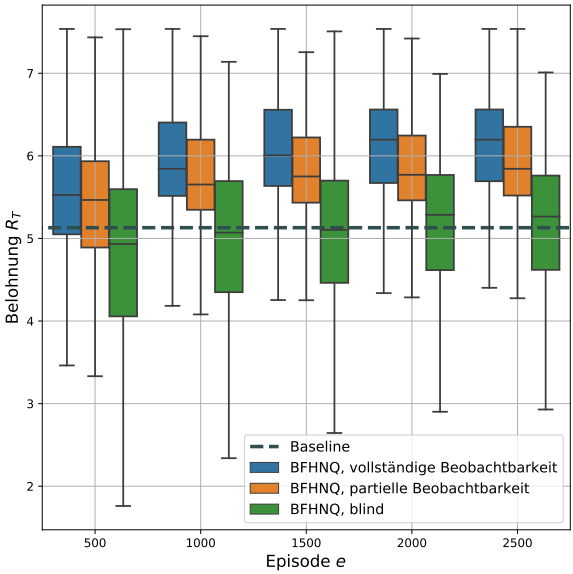



Abbildung 4.8: Verteilungen der *online* erhaltenen Belohnung für unterschiedliche Beobachtbarkeits-Szenarien, gruppiert für jeweils 500 aufeinanderfolgende Episoden $e$ als Box-Plot und rechnerisch ermittelter Erwartungswert der *Baseline*-Belohnung.

auch mit niedrigen Explorationsraten schnell konvergiert. Die Lernrate $\alpha$ hat nur einen sehr geringen Einfluss auf die Ergebnisse des BFHNQ Algorithmus für den betrachteten Anwendungsfall. Mittelwert $\hat{r}_\mu$ und Standardabweichung $\hat{r}_\sigma$ der beobachteten Belohnung während der letzten 250 Episoden mit einer Explorationsrate von $\varepsilon_0 = 0.3$ betragen für $\alpha = 0.5$ $(\hat{r}_\mu = 5.87, \hat{r}_\sigma = 0.63)$, für $\alpha = 0.7$ $(\hat{r}_\mu = 5.89, \hat{r}_\sigma = 0.78)$ und für $\alpha = 0.9$ $(\hat{r}_\mu = 5.90, \hat{r}_\sigma = 0.69)$.
Neben der Parameter-Studie wurden Experimente durchgeführt, um den Effekt der partiellen Prozess-Beobachtbarkeit zu untersuchen. Für drei unterschiedliche Szenarien wurden je 10 unabhängige Optimierungsläufe ausgeführt. Ergebnisse der Untersuchung sind in Abbildung 4.8 dargestellt. Im Szenario

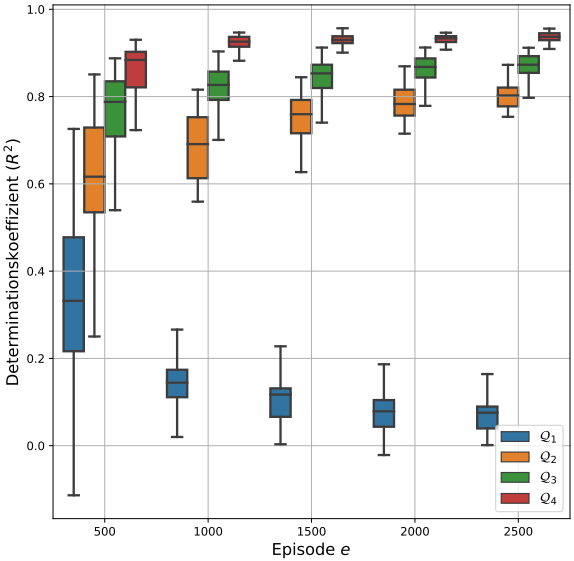



Abbildung 4.9: Verteilungen der Bestimmtheitsmaß-Werte ($R^2$) der $\mathcal{Q}_t$-Netze für das Szenario *partielle Beobachtbarkeit*, ermittelt durch 5-fache Kreuzvalidierung und gruppiert für jeweils 500 aufeinanderfolgende Episoden $e$ als Box-Plot.

*vollständige Beobachtbarkeit* (blaue Boxen) sind die aktuellen Prozessbedingungen in Form des Reibungskoeffizienten für den Agenten als Teil der Zustandsbeschreibung einsehbar. Das Szenario *partielle Beobachtbarkeit* (orange Boxen) entspricht dem vorgestellten und im Vorangegangenen untersuchten Standard-Szenario. Im Szenario *keine Beobachtbarkeit* (grüne Boxen) ist der Agent „blind“ und erhält keine beobachtbaren Größen. Hierbei ist die durch den Agenten während des Lernens erzielte Belohnung in Form eines kombinierten Box-Plots dargestellt. Eine Box stellt die Verteilung der erhaltenen Belohnungen für das entsprechende Szenario über die 10 unabhängigen Optimierungsläufe und 500 aufeinanderfolgende Episoden dar.
Die Güte der $\mathcal{Q}$-Netze wurde jeweils nach dem Training der Netze durch 5-fache Kreuzvalidierung ermittelt. Zur Bewertung der Funktionsapproximationen

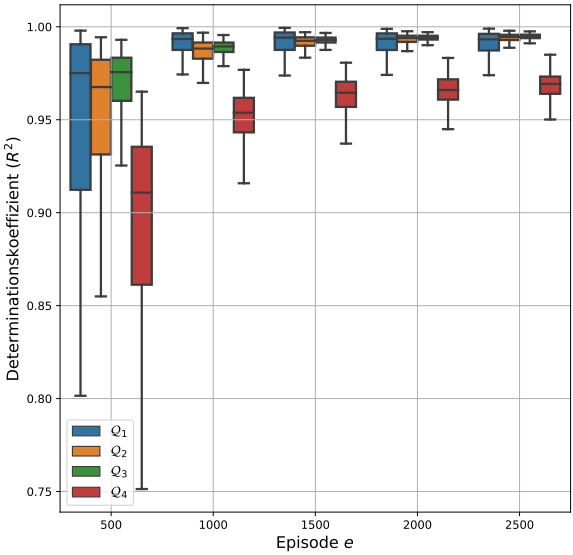



Abbildung 4.10: Verteilungen des Bestimmtheitsmaß-Werte ($R^2$) der $\mathcal{Q}_t$-Netze für das Szenario *vollständige Beobachtbarkeit*, ermittelt durch 5-fache Kreuzvalidierung und gruppiert für jeweils 500 aufeinanderfolgende Episoden $e$ als Box-Plot.

$\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3, \mathcal{Q}_4$ wird das mittlere Bestimmtheitsmaß ($R^2$) der Kreuzvalidierung genutzt. In Abbildung 4.9 ist dieser für einen 2500 Episoden andauernden Lernvorgang des BFHNQ Algorithmus über die Episoden-Sequenz aufgetragen. Dabei werden jeweils die mittleren Werte des Bestimmtheitsmaßes aus 10 aufeinanderfolgenden Trainingsphasen (entspricht 500 Episoden, da $n_{\mathcal{Q}} = 50$) zu einer Verteilung zusammengefasst. Dargestellt sind diese Verteilungen pro Modell $\mathcal{Q}_t$ jeweils als Box-Plot. Für die Modelle $\mathcal{Q}_2, \mathcal{Q}_3, \mathcal{Q}_4$ steigt das Bestimmtheitsmaß wie erwartet, aufgrund der anwachsenden Trainingsdatenmenge, an. Das Bestimmtheitsmaß für die Approximation der Q-Werte für den ersten Kontrollschritt $\mathcal{Q}_1$ weist jedoch einen negativen Verlauf auf. Es konnten keine Hyperparameter der $\mathcal{Q}$-Netze gefunden werden die zu einem abweichenden Verhalten führen.

Grundlegend anders verhält es sich im Fall vollständiger Beobachtbarkeit, in dem der Reibungskoeffizient durch den Agenten als Teil der beobachtbaren Werte gegeben ist. Die Werte des Bestimmtheitsmaßes sind für diesen Fall in Abbildung 4.10 auf gleiche Art dargestellt. Hier nähert sich der Wert des mittleren Bestimmtheitsmaßes $\mathcal{Q}_1$ sehr schnell 1 an, was darauf hinweist, dass der negative Verlauf mit der partiellen Beobachtbarkeit zusammenhängt. Dieser Zusammenhang wird in Abschnitt 4.6 eingehender betrachtet.

### 4.4.2 Untersuchung der Dateneffizienz im deterministischen Fall

Eine wesentliche Eigenschaft eines Verfahrens zum Lernen optimaler Regelungsstrategien für Fertigungsprozesse ist die Dateneffizienz. Durch die Notwendigkeit der Exploration während des Lernens entstehen in einer Surrogat-Umgebung Rechenkosten für die Simulation des Prozesses und in einer physikalischen Prozessumgebung Kosten durch erhöhte Fehlproduktionsraten. Die Dateneffizienz des BFHNQ Algorithmus wurde für eine deterministische Variante des Tiefziehprozesses, vergleichend mit einem klassischen Hill-Climbing Ansatz, untersucht. Grundlage ist der in Abschnitt 4.3 vorgestellte Prozess. Der Reibungskoeffizient wurde allerdings, abweichend zu den bisher geschilderten Untersuchungen der Beschreibung in 4.3.2, als konstant $\mu = 0.028$ angenommen. Zur vollständigen Beschreibung des Zustandes genügt dann die Abfolge der bisherigen Aktionen der aktuellen Episode. Die Observablen (siehe 4.3.3) entfallen und Zeile 10 aus Listing 8 wird ersetzt durch $\tilde{\mathbf{s}}' \leftarrow \tilde{\mathbf{s}} \frown a$. Für die Untersuchungen wurde die in 4.3.4 eingeführte Belohnungsfunktion mit $\mathbf{w} = (0.25, 0.25, 0.5)^\intercal$ für $\mathfrak{r} = (\mathfrak{r}_{\text{mises}}, \mathfrak{r}_{\text{wand}}, \mathfrak{r}_{\text{verbrauch}})^\intercal$ gewichtet.
Als Grundlage der Untersuchungen wurden für den Prozess alle wählbaren Prozesspfade für $\mu = 0.028$ simuliert. Die Pfade sind in Abbildung 4.11 jeweils als Verlauf der Niederhaltekraft $F_{nh}$ (siehe 4.3.1) abgebildet, wobei die Farbe des abgebildeten Pfads der Belohnung am Prozessende entspricht. Die vier,

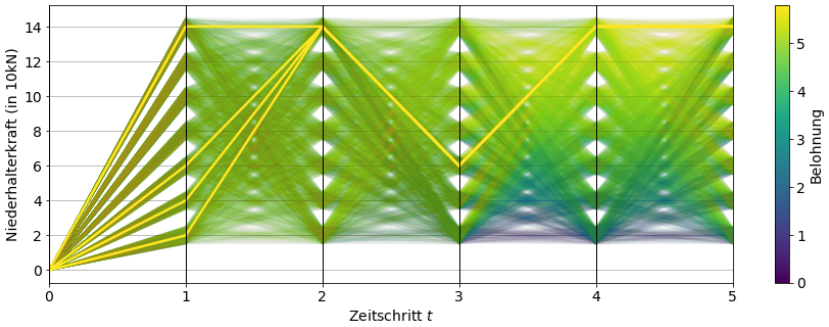



Abbildung 4.11: Wert der in 4.3.4 eingeführten Belohnungsfunktion mit Gewichten $\mathbf{w} = (0.25, 0.25, 0.5)^\intercal$ für alle $7^5$ möglichen Prozesspfade des Prozesses mit einem Reibungskoeffizient von $\mu = 0.028$. Pro Prozesspfad ist die Höhe der Belohnung durch die Farbe repräsentiert. Die drei Prozesspfade mit den höchsten Werten des jeweiligen Belohnungsterms sind mit einem Alphawert von 1 dargestellt, alle weiteren mit einem Alphawert von 0.01. Prozesspfade sind absteigend nach Höhe der Werte des jeweiligen Belohnungsterms sortiert.

bezüglich der Belohnungsfunktion, besten Prozesspfade sind in der Abbildung hervorgehoben.

Zur empirischen Quantifizierung der Dateneffizienz wurde untersucht, wie viele Prozesspfade durch den jeweiligen Algorithmus abgetastet werden, bis der beste Prozesspfad ermittelt wird und wie viele Abtastungen notwendig sind um einen der vier hervorgehobenen Prozesspfade zu ermitteln. Vergleichend untersucht wurden dabei:

1. BFHNQ wie beschrieben, mit den in 4.3.6 spezifizierten Parametern und Netzarchitekturen.

2. Einem *Steepest Ascent random-Restart Hill Climbing* Ansatz, der durch einen zufälligen Pfad initialisiert wird, in jedem Schritt alle benachbarten Prozesspfade evaluiert und den besten benachbarten Pfad als Ausgang für den nächsten Optimierungsschritt nutzt. Sobald ein lokales Optimum

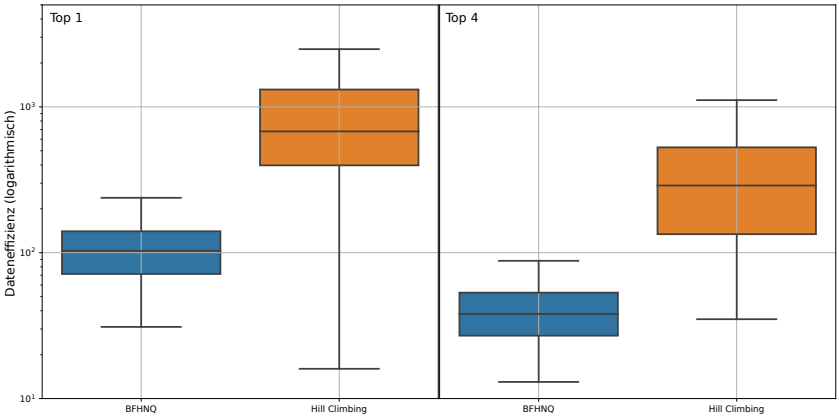



Abbildung 4.12: Verteilungen der benötigten Samples in jeweils 100 unabhängigen Optimierungsläufen zur Ermittlung eines gesuchten Optimums (logarithmisch). Links: benötigte Samples zur Ermittlung des optimalen Prozesspfades bezüglich der Belohnungsfunktion. Rechts: benötigte Samples zur Ermittlung einer der vier besten Prozesspfade bezüglich der Belohnungsfunktion.

erreicht ist, dass keinem der gesuchten Pfade entspricht, wird der Algorithmus mit einem zufällig gewählten Prozesspfad neu initialisiert.

Die Verteilung der Anzahl der erprobten Prozesspfade über jeweils 100 unabhängige Optimierungsläufe ist für beide Methoden in Abbildung 4.12 als kombinierter Box-Plot mit logarithmischer $y$-Achse dargestellt. Im Mittel wurden durch BFHNQ 109.72 Prozesspfade abgetastet um den optimalen Prozesspfad zu ermitteln und 42.41 Prozesspfade abgetastet um einen der vier besten Prozesspfade zu finden. Der *Hill Climbing* Ansatz benötigt hierfür im Mittel 951.82 beziehungsweise 399.76 Samples. Exemplarische Optimierungsläufe der beiden Ansätze sind in Anhang B.0.1 dargestellt.

## 4.5 Multikriterielle Erweiterung

Die Formulierung der Prozessoptimierung als Markov-Entscheidungsprozess umfasst die Definition einer skalaren Belohnungsfunktion. Häufig werden hierzu, wie in 4.3.4, mehrere Belohnungsterme $\mathfrak{r}_l$ kombiniert, indem eine Skalarisierungsfunktion $\mathfrak{f}$ angewandt wird

$$R(s) = \mathfrak{f}(\mathfrak{r}, \mathbf{w}). \tag{4.11}$$

Gewichtswerte $w_l \in \mathbb{R}$ werden dabei genutzt, um einzelne Belohnungskriterien $\mathfrak{r}_l$ zu priorisieren. Der Gewichtsvektor $\mathbf{w} \in \mathbb{R}^n$, für den gilt $w_i \geq 0 \forall i, \|\mathbf{w}\|_1 = 1$, wird im Folgenden auch als Konfiguration der Belohnungsfunktion bezeichnet. Einzelne Belohnungskriterien sind oft gegenläufig und die Konfiguration ändert sich in einigen Anwendungen über die Zeit. So hängt beispielsweise die Abwägung von Materialverbrauch und Produktqualität von zeitlich variablen wirtschaftlichen Rahmenbedingungen (bspw. dem Materialwert) und anwendungsbezogenen Rahmenbedingungen (bspw. den für den Anwendungskontext spezifizierten Qualitätseigenschaften des Produkts). In der Praxis wird die aktuelle Konfiguration manuell oder durch einen Algorithmus auf der Prozessplanungs-Ebene vorgegeben. Da die skalare Belohnungsfunktion $R$ von der Konfiguration abhängt, sind gelernte Erwartungswertfunktionen der zukünftigen Belohnung (Q-Funktionen) durch die Änderung der Konfiguration hinfällig. Eine Möglichkeit, dem zu begegnen, ist das Lernen einer multi-kriteriellen, vektorwertigen Q-Funktion $\mathcal{Q}^{MK}$, wobei $\mathcal{Q}_i^{MK}$ den Erwartungswert bezüglich des Belohnungsterms $\mathfrak{r}_i$ darstellt [41].

### 4.5.1 Erweiterungsansatz

In diesem Abschnitt wird eine Erweiterung des BFHNQ Algorithmus behandelt, die es ermöglicht, auch in dem skizzierten Fall zeitlich variierender Konfigurationen der Belohnungsfunktion mit linearer Skalarisierungsfunktion effizient zu lernen. Ergebnisse der Untersuchung des erweiterten Algorithmus für den in Abschnitt 4.3 beschriebenen Anwendungsfall werden anschließend erläutert.

Anstelle des skalarisierten Belohnungssignals $r$ erhält der BFHNQ-Agent nun den Belohnungsvektor $\mathfrak{r} \in \mathbb{R}^n$ und lernt eine vektorwertige Q-Funktion $\mathcal{Q}^{MK}$, wobei $\mathcal{Q}_i^{MK}$ dem approximierten Erwartungswert der zukünftigen Belohnung $\mathfrak{r}_i$ entspricht. Die Skalarisierungsfunktion $\mathfrak{f}$ und die aktuelle Konfiguration $\mathbf{w}$ wird als bekannt angenommen. Für ein multikriterielles Erfahrungs-Tupel $(\tilde{\mathbf{s}}, a, \tilde{\mathbf{s}}', \mathfrak{r})$ ist die $i$-te Komponente $y_{\mathcal{Q}^{MK},i}$ der Zielgröße des $\mathcal{Q}^{MK}$-Netzes durch Übertragung der *Q-Learning Update* Regel aus (4.3) für $\mathcal{Q}_t = \mathcal{Q}_t(\tilde{\mathbf{s}}, a, \theta_t)$ gegeben als

$$\mathbf{y}_{\mathcal{Q}^{MK},i} = \mathcal{Q}_{t,i}^{MK} + \alpha\left(\mathfrak{r}_i + \gamma \max_{a' \in A} \mathcal{Q}_{t+1,i}^{MK}(\tilde{\mathbf{s}}', a', \theta_{t+1}) - \mathcal{Q}_{t,i}^{MK}\right) \tag{4.12}$$

Bei dieser Form des Updates wird $a'$ für jeden Term $\mathfrak{r}_i$ unabhängig voneinander ermittelt. Dies ist problematisch und führt zu einem systematischen Verzerrungseffekt, da hierbei vorausgesetzt wird, dass die Zielstrategie für den jeweiligen Term, unabhängig von den anderen Termen, optimale Entscheidungen trifft, was die vorhandenen, wechselseitigen Abhängigkeiten vernachlässigt.

Um dies zu vermeiden, werden bei multikriteriellen Verfahren üblicherweise *on-Policy* Updates verwendet. Das entsprechende *SARSA Update* (siehe 2.2.6) für das Erfahrungs-Tupel $(\tilde{\mathbf{s}}, a, \mathfrak{r}, \tilde{\mathbf{s}}', a')$ ist definiert als

$$\mathbf{y}_{\mathcal{Q}^{MK}} = \mathcal{Q}_t^{MK} + \alpha\left(\mathfrak{r} + \gamma \mathcal{Q}_{t+1}^{MK}(\tilde{\mathbf{s}}', a', \theta_{t+1}) - \mathcal{Q}_t^{MK}\right). \tag{4.13}$$

Da das hierfür verwendete *Replay Memory* $\mathcal{D}$ aus historischen Erfahrungsdaten besteht, wurde die Folgeaktion $a'$ nicht der aktuellen Lernstrategie $\tilde{\pi}$ folgend ausgeführt, sondern entstammt der Lernstrategie zum Zeitpunkt der Erfahrung. Aus diesem Grund werden für das *SARSA Update* die historischen Daten $(\tilde{\mathbf{s}}, a, \mathfrak{r}, \tilde{\mathbf{s}}')$ aus $\mathcal{D}$ genutzt und $a'$ während des Updates unter Berücksichtigung der aktuellen Lernstrategie $\tilde{\pi}$ erzeugt[5].
Zur Bestimmung der *greedy*-Strategie $\pi$, und der davon abgeleiteten $\varepsilon$-*greedy* Lernstrategie $\tilde{\pi}$ für eine gegebene Konfiguration $\mathbf{w}$ wird die Skalarisierungsfunktion $\mathfrak{f}$ auf den Vektor der Term-weisen erwarteten Belohnungen angewandt

$$\pi(\mathbf{s}_t, \mathbf{w}) = \arg\max_{a \in A} \mathfrak{f}\big(\mathcal{Q}_t^{MK}(\mathbf{s}_t, a, \theta_{t+1}), \mathbf{w}\big). \tag{4.14}$$

Dabei wird angenommen, dass folgende Gleichung für alle $s$ und $\mathbf{w}$ erfüllt ist

$$\arg\max_{a \in A} \Big(\mathfrak{f}\big(Q_{t,\pi}^{MK}(s,a), \mathbf{w}\big)\Big) = \arg\max_{a \in A} \big(Q_{t,\pi}(s,a)\big), \tag{4.15}$$

wobei $Q_{t,\pi}^{MK}$ eine tabellarische vektorwertige Q-Funktion der Strategie $\pi$ darstellt und die tabellarische skalare Q-Funktion $Q_{t,\pi}$ Erwartungswerte bezüglich der skalaren Belohnung $R = \mathfrak{f}(\mathfrak{r}, \mathbf{w})$ für $\pi$ angibt. Diese Annahme trifft nur für lineare Skalarisierungsfunktionen $\mathfrak{f}$ zu (siehe [97]). In Fällen mit nicht-linearer Skalarisierungsfunktion sind der beschriebene Ansatz und darauf basierende Algorithmen grundsätzlich als Heuristiken anzusehen.
Die im Folgenden dargestellten Untersuchungen der beschriebenen Erweiterung des BFHNQ Algorithmus für Fälle mit variabler Konfiguration basiert auf einer Variante des in Abschnitt 4.3 beschriebenen Optimierungsproblems. Die verwendete nicht-lineare Skalarisierungsfunktion entspricht der in (4.10) definierten Belohnungsfunktion $R$

[5] Wie sich bei der Analyse der Ergebnisse herausgestellt hat und unten weiter ausgeführt wird erzeugt auch die Verteilung der Aktionen $a$ in den historischen Daten beim *on-Policy* Update einen Bias.

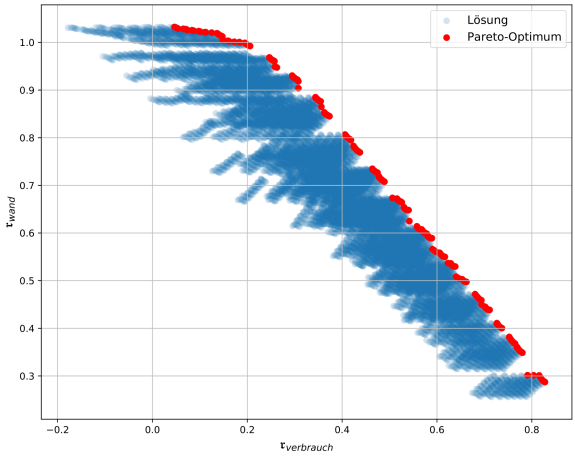



Abbildung 4.13: Erzielte Belohnungsterme $\mathfrak{r}_{\text{wand}}, \mathfrak{r}_{\text{verbrauch}}$ pro Lösung für den Reibungskoeffizienten $\mu = 0.028$. Rote Punkte repräsentieren *Pareto-Optima* für $\mu = 0.028$. Blaue Punkte sind mit einem Alphawert von 0.2 dargestellt und repräsentieren dominierte Lösungen.

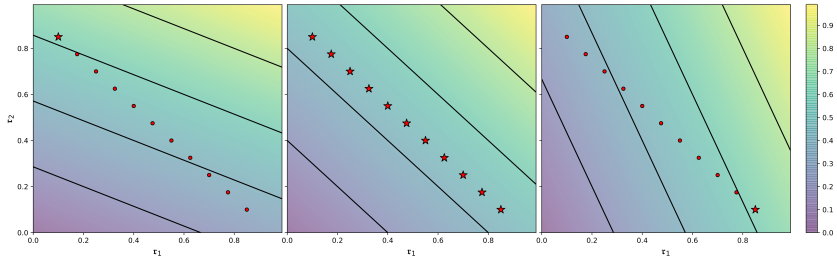
$$\mathfrak{f}(\mathfrak{r}, \mathbf{w}) = \begin{cases} 10 \times H(\mathfrak{r}, \mathbf{w}) & \text{, if } \min(\mathfrak{r}) > 0, \\ 0 & \text{, else,} \end{cases} \tag{4.16}$$

wobei $\mathfrak{r} = (0,0)^\intercal$ für $t < T$ gilt. Der Diskontierungsfaktor beträgt $\gamma = 1$. Bei den Untersuchungen werden lediglich zwei der drei Belohnungsterme betrachtet, um das Optimierungsverhalten in zwei-dimensionalen Diagrammen darstellen zu können. Berücksichtigt werden die Belohnungsterme $\mathfrak{r}_{\text{wand}}$ und $\mathfrak{r}_{\text{verbrauch}}$ während der Belohnungsterm $\mathfrak{r}_{\text{mises}}$ bei den Untersuchungen keine Rolle spielt. Die insgesamt $7^5$ unterschiedlichen Ausführungen des Tiefziehprozesses mit einen festen Reibungskoeffizienten von $\mu = 0.028$ wurden vollständig simuliert und sind in Abbildung 4.13 dargestellt. In der Abbildung sind pro Lösung die erhaltenen Belohnungen der Terme $(\mathfrak{r}_{\text{wand}}, \mathfrak{r}_{\text{verbrauch}})$ dargestellt. Diese

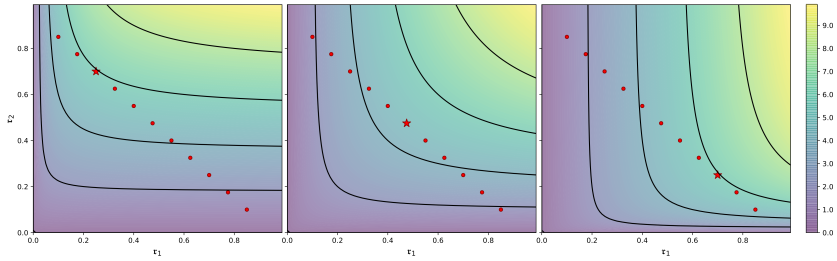
Darstellung ermöglicht eine Abschätzung des Lösungsraumes und eine Annäherung der *Pareto-Front* bezüglich $(\mathfrak{r}_{\text{wand}}, \mathfrak{r}_{\text{verbrauch}})$. Rot dargestellte Lösungen liegen auf der so angenäherten *Pareto-Front* und werden im Folgenden als *Pareto-Optima* bezeichnet. Ein *Pareto-Optimum* zeichnet aus, dass kein besseres Ergebnis bezüglich eines einzelnen Terms $\mathfrak{r}_i$ existiert, das nicht gleichzeitig in Bezug auf einen anderen Term $\mathfrak{r}_j, j \neq i$ eine Verschlechterung darstellt. Die Gesamtheit der *Pareto-Optima* wird als *Pareto-Front* bezeichnet. Abbildungen in Form von 4.13 für weitere Reibungskoeffizienten sind in Anhang B.0.2 dargestellt.
Bei der Darstellung der Lösungsmenge für $\mu = 0.028$ in Abbildung 4.13 und für weitere Reibungskoeffizienten in Anhang B.0.2 fällt auf, dass ein großer Teil ($\mathfrak{r}_{\text{verbrauch}} > 0.2$) der *Pareto-Front* nahezu linear verläuft. Wie in Abbildung 4.14 (a) dargestellt, gilt bei einem linearen Verlauf der *Pareto-Front* für eine lineare Skalarisierungsfunktion $\mathfrak{f} = w_1\mathfrak{r}_1 + w_2\mathfrak{r}_2$ mit $w_1 + w_2 = 1$, dass

- für eine bestimmte Konfiguration $\mathbf{w} = (w_1, w_2)$ alle auf der *Pareto-Front* liegenden Lösungen bezüglich $\mathfrak{f}$ gleichwertig sind (mittlerer Fall in Abbildung 4.14 (a)),
- für alle Konfigurationen $\mathbf{w}' = (w_1', w_2')$ mit $w_1' > w_1$ die Lösung mit maximalem Wert $\mathfrak{r}_1$ das Optimum bezüglich $\mathfrak{f}$ darstellt (linker Fall in Abbildung 4.14 (a)) und
- für alle Konfigurationen $\mathbf{w}' = (w_1', w_2')$ mit $w_2' > w_2$ die Lösung mit maximalem Wert $\mathfrak{r}_2$ das Optimum bezüglich $\mathfrak{f}$ darstellt (rechter Fall in Abbildung 4.14 (a)).

*Pareto-Optima*, die in konkaven Bereichen einer *Pareto-Front* liegen sind für keine Konfiguration der linearen Skalarisierungsfunktion $\mathfrak{f}$ optimal.
Anders verhält es sich bei konkaven Skalarisierungsfunktionen wie der Funktion in (4.16). Für einen linearen Verlauf der *Pareto-Front* ist diese in ist in Abbildung 4.14 (b) dargestellt. Für jede Konfiguration existiert in dem skizzierten



(a) Lineare Skalarisierung $\mathfrak{f} = w_1\mathfrak{r}_1 + w_2\mathfrak{r}_2$



(b) Konkave Skalarisierung $\mathfrak{f}\big((\mathfrak{r}_1,\mathfrak{r}_2)^\intercal,(w_1,w_2)^\intercal\big)$ nach (4.13)

Abbildung 4.14: Optima in Bezug auf die lineare Skalarisierungsfunktion (a) und eine konkave Skalarisierungsfunktion (b) für einen linearen Verlauf der *Pareto-Front*. *Pareto-Optima* sind als Punkte und Sterne abgebildet. Durch Sterne sind *Pareto-Optima* gekennzeichnet, die bezüglich $(\mathfrak{f},\mathbf{w})$ optimal sind. Dargestellt sind jeweils von links nach rechts (a) die lineare Skalarisierungsfunktion in den Konfigurationen $\mathbf{w} = (0.3,0.7)^\intercal$ $\mathbf{w} = (0.5,0.5)^\intercal$, $\mathbf{w} = (0.7,0.3)^\intercal$ und (b) die konkave Skalarisierungsfunktion in den Konfigurationen $\mathbf{w} = (0.1,0.9)^\intercal$, $\mathbf{w} = (0.5,0.5)^\intercal$, $\mathbf{w} = (0.9,0.1)^\intercal$. Die konfigurierte Funktion $\mathfrak{f}(\bullet,\mathbf{w})$ ist jeweils durch den Farbverlauf und vier äquidistante Isolinien dargestellt.

Fall ein *Pareto-Optimum* und für jedes *Pareto-Optimum* auf der linearen *Pareto-Front* existiert eine Konfiguration von $\mathfrak{f}$ für die das *Pareto-Optimum* optimal ist.
Die Ungenauigkeit der vorgestellten Heuristik aufgrund der nicht zutreffenden Annahme 4.15 wächst mit zunehmender Streuung der Zustandsübergänge des Entscheidungsproblems. Um die Effektivität der heuristisch erweiterten Methode zeigen zu können, ist die Streubreite des Reibungskoeffizienten und der beobachtbaren Größen für die hier vorgestellten Experimente gegenüber den vorangegangen beschriebenen Experimenten reduziert. Parameter der Beta-Verteilung des Reibungskoeffizienten sind $(p_\beta = 3, q_\beta = 15)$. Das additive Gaußsche Rauschen der beobachtbaren Größen wurde mit $\sigma = 0.5\%$ für $F_{\text{stempel}}, u_{\text{blech}}$ und $\sigma = 0.25\%$ für $v_{\text{nh}}$ modelliert.

## 4.5.2 Untersuchung und Ergebnisse

Zur Untersuchung der multikriteriellen Erweiterung wurden aus vier Stufen *a*, *b*, *c*, *d* bestehende Experimente durchgeführt. Jede Stufe entspricht dabei einem $n_e = 1000$ Lernepisoden andauernden Optimierungslauf mit individueller Konfiguration $\mathbf{w}$. Die gelernte vektorwertige Q-Funktion $\mathcal{Q}^{\text{MK}}$ wird auf die jeweils Nächste Stufe, in Form der Zeitschritt-abhängigen *Replay Memories* $\mathcal{D}_t$ übertragen (vgl. 4.2). Die Konfiguration $\mathbf{w}$ wird pro Optimierungslauf unabhängig zufällig aus der Menge $\{(0.1,0.9)^\intercal, (0.2,0.8)^\intercal, ..., (0.9,0.1)^\intercal\}$ gezogen. Pro Stufe wird BFHNQ, Listing 8 folgend, ausgeführt. Zusätzliche Parameter sind die initialen *Replay Memories* $\mathcal{D}_t$, die Skalarisierungsfunktion $\{$, sowie die Konfiguration $\mathbf{w}$. Die Initialisierung der *Replay Memories* (Listing 8, Zeilen 2 bis 4) entfällt. Die Lernstrategie $\tilde{\pi}$ ist $\varepsilon$*-greedy*, folgend (4.14) und anstelle des skalaren Belohnungssignals $r$ wird das vektorwertige Belohnungssignal $\mathfrak{r}$ beobachtet (Listing 8, Zeile 9). Die Zielgrößenberechnung beim Lernen der $Q^{\text{MK}}$-Modelle geschieht entweder durch das multikriterielle *Q-Learning*

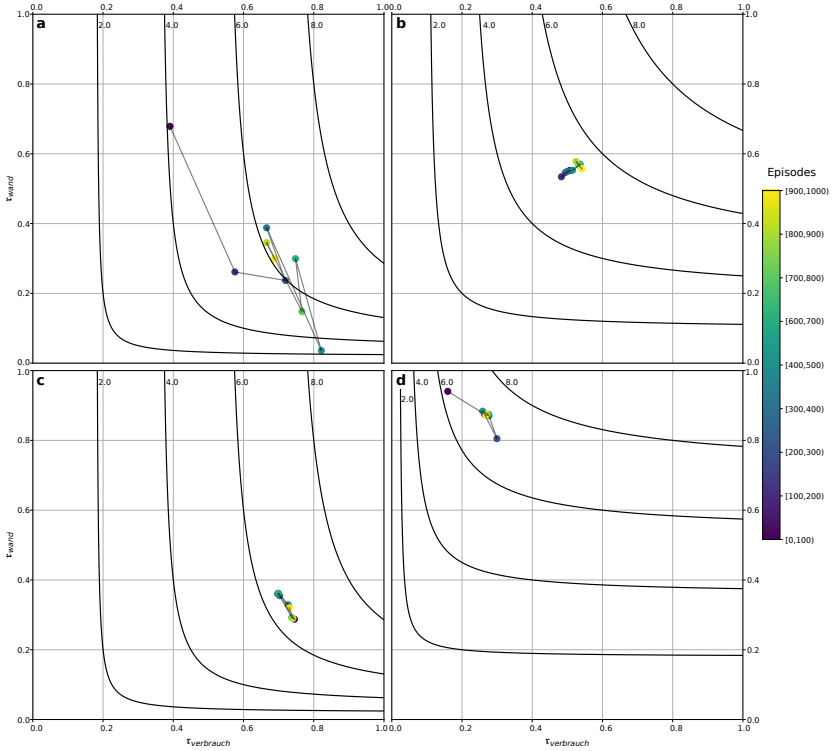



Abbildung 4.15: Verlauf eines Experiments mit wechselnden Konfigurationen. Für vier aufeinanderfolgende Stufen (von *a* oben links bis *d* unten rechts) mit jeweils eigenen Konfigurationen $(((0.9,0.1)^\intercal,(0.5,0.5)^\intercal,(0.9,0.1)^\intercal,(0.1,0.9)^\intercal))$: Isolinien der Skalarisierungsfunktion in Abhängigkeit der Konfiguration und Optimierungsverlauf über jeweils 1000 aufeinanderfolgende Episoden, im Raum der Belohnungsterme $\tau_{verbrauch}$, $\tau_{wand}$

*Update* (4.12) oder durch das multikriterielle *SARSA Update* (4.13) (Listing 8, Zeile 18). MORL Parameter sind $\varepsilon_0 = 0.1, \lambda_\varepsilon = 10^{-3}, \alpha = 0.7, n_\mathcal{Q} = 50$.
Der Verlauf eines einzelnen Experiments ist in Abbildung 4.15 dargestellt. Die Konfiguration $\mathbf{w} = (w_{\text{verbrauch}}, w_{\text{wand}})^\intercal$ der Belohnungsfunktion entspricht dabei für die Stufen *a* bis *d* $((0.9,0.1)^\intercal,(0.5,0.5)^\intercal,(0.9,0.1)^\intercal,(0.1,0.9)^\intercal)$. Die

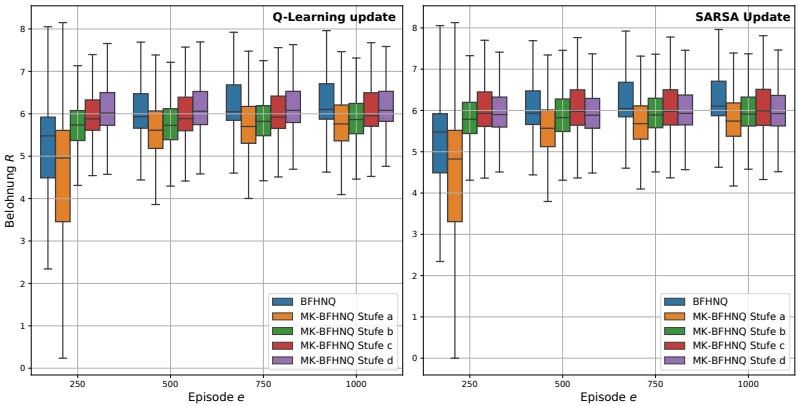



Abbildung 4.16: Verteilungen der erhaltenen Belohnung $R$ pro Episode $e$ für BFHNQ mit skalaren $\mathcal{Q}_t$-Netzen und dem hier beschriebenen *MK-BFHNQ* Ansatz mit *Q-Learning Update* (links) und *SARSA Update* (rechts) für vier Stufen. Ein Box-Plot zeigt die Verteilung der erhaltenen Belohnungen für 250 aufeinanderfolgende Episoden und 100 unabhängig durchgeführte Experimente.

konfigurierte Skalarisierungsfunktion ist pro Stufe in Form der 2.0-,4.0-,6.0- und 8.0-Isolinien dargestellt. Die 0.0-Isolinie liegt unabhängig von der Konfiguration auf den Koordinatenachsen, da $\mathfrak{f}(\mathfrak{r}, \mathbf{w}) = 0$ für $\min(\mathfrak{r}) \leq 0$. Der Verlauf der Optimierung ist Pro Stufe durch eine Folge von 9 Punkten dargestellt, wobei die Koordinaten der Punkte jeweils die mittlere erhaltene Belohnung pro Term $\mathfrak{r}$ in 100 aufeinanderfolgenden Episoden angeben.

Zur quantitativen Untersuchung der multikriteriellen Erweiterung wurden die oben beschriebenen Experimente wiederholt durchgeführt. Stochastische Größen, insbesondere die Konfiguration pro Stufe und der Reibungskoeffizient pro Episode, wurden dabei unabhängig zufällig gezogen. Um zu untersuchen wie sich der systematische Verzerrungseffekt im Fall des *off-Policy Q-Learning Updates* (4.12) auswirkt und wie es sich im direkten Vergleich zu dem *on-Policy* Update (4.13) verhält, wurden beide Varianten evaluiert. Ebenso wurde der BFHNQ Algorithmus wie in 4.2 vorgestellt, mit skalarer Belohnungsfunktion

und skalaren *Q-Netzen*, unter den hier beschriebenen Prozessbedingungen und zufällig gewählter Konfiguration der Belohnungsfunktion als *Baseline* evaluiert. Für jedes dieser Experimente wurden 100 unabhängige Wiederholungen durchgeführt.

Abbildung 4.16 stellt Ergebnisse der Experimente mit dem *Q-Learning Update* (links) und dem *SARSA Update* (rechts) dar. Dabei ist jeweils die Verteilung der über die 100 Wiederholungen in 250 aufeinanderfolgenden Episoden erhaltenen skalarisierten Belohnung $R_T$ bzw. ϱ zum Zeitpunkt $T$ als Box-Plot dargestellt. Ergebnisse der Experimente mit dem multikriteriellen Ansatz sind für die vier Stufen *a* bis *d* getrennt dargestellt. Ergebnisse pro Stufe, sowie Ergebnisse des skalaren BFHNQ-Ansatzes (blau) sind farblich voneinander getrennt.

Abbildung 4.15 zeigt das Verhalten der multikriteriellen Erweiterung anhand eines einzelnen Experiments. Der Verlauf der Optimierung in Stufe *a* weist im Vergleich zu den nachfolgenden Stufen eine sehr hoher Varianz der Belohnungen ϱ auf. Insbesondere in Stufe *b* und *c* befindet sich der Ausgangspunkt sehr nah an der gefundenen Lösung. Die Konfiguration der Belohnungsfunktion für die Stufen *a* und *c* sind identisch. Die in Stufe *c* gefundene Lösung ist der in Stufe *a* gefundenen Lösung deutlich überlegen. Diese Charakteristika spiegeln sich auch in den in Abbildung 4.16 dargestellten quantitativen Ergebnissen wieder. Die Statistiken der erreichten skalaren Belohnungen in der initialen Stufe *a* ist für beide Update-Formen deutlich niedriger wie die durch skalares BFHNQ erreichten Belohnungssignale. Für beide Update-Formen ist ein positiver Effekt der Übertragung des Prozesswissens früherer Konfigurationen beobachtbar. Ab Stufe *b* sind beide Varianten während der ersten 250 Episoden dem skalaren Ansatz deutlich überlegen. Im weiteren Optimierungsverlauf gleichen sich die Ergebnisse der Ansätze an. Gegen Ende der 1000 Episoden sind die mittleren erhaltenen Belohnungen des skalaren Ansatzes höher als die durch die multikriteriellen Ansätze in den jeweils besten Stufen erreichten Ergebnisse. Die mittlere Belohnung der letzten 250 Episoden beträgt für skalares BFHNQ 6.27, für die Variante mit *Q-Learning Update* in Stufe *d*: 6.18 und für die Variante mit SARSA Update in Stufe *c*: 6.11.

## 4.6 Diskussion der Ergebnisse

Im Folgenden werden die in diesem Kapitel vorgestellten Ergebnisse der Untersuchungen im Fall stochastischer Prozessbedingungen 4.4.1, im Fall eines deterministischen Prozesses 4.4.2 und im Fall der multikriteriellen Entscheidungsoptimierung 4.5.2 zusammenfassend bewertet. Eine Diskussion der Algorithmen und Ergebnisse im Gesamtkontext der Arbeit findet in Kapitel 6.1 statt.

In 4.4.1 werden Untersuchungsergebnisse des BFHNQ Algorithmus für den partiell beobachtbaren Tiefziehprozesses unter variierenden Prozessbedingungen vorgestellt. BFHNQ wird dabei mit einem hypothetischen Modellbasierten Ansatz verglichen, der auf einem Prozessmodell beruht, das unter der Annahme statischer Prozessbedingungen erstellt wurde. Abbildung 4.7 zeigt, dass BFHNQ mit zunehmender Erfahrung aus den Episoden immer bessere Strategien lernt. Bereits nach etwa 150 Episoden übertrifft es das Baseline-Verfahren und erreicht im Mittel eine ca. 18% höhere Belohnung, welche hier die Qualität der Prozessresultate widerspiegelt. Dargestellt wird die erwartete Belohnung, wenn die aktuell gelernte Strategie ohne Exploration ausgeführt wird. Das Rauschen durch die $\epsilon$-*greedy* Lernstrategie wird dadurch, im Unterschied zu den darauffolgenden Abbildungen der erhaltenen Belohnung, nicht abgebildet. Untersuchungen verschiedener Beobachtbarkeits-Szenarien sind in Abbildung 4.8 dargestellt. Erwartungskornform erzielt BFHNQ unter vollständiger Beobachtbarkeit bessere Ergebnisse als in dem Standard-Szenario der partiellen Beobachtbarkeit, da der Agent den Reibungskoeffizienten einsehen kann und so vom ersten Zeitschritt einer Episode an die Aktionen auf die aktuellen Prozessbedingungen abstimmen kann. Im Fall ohne beobachtbare Größen kann keine Anpassung an den Reibungskoeffizienten erfolgen. Folge sind deutlich schlechtere Ergebnisse und eine hohe Streuung der Ergebnisse. Der Erwartungswert ist hierbei nach einigen Episoden vergleichbar mit der modellbasierten Baseline, die aufgrund des eingeschränkten Modells ebenfalls nicht in der Lage ist, Aktionen in Abhängigkeit der Prozessbedingungen zu wählen.

Untersuchungsergebnisse zu der Qualität der einzelnen $\mathcal{Q}$-Netze für den partiell beobachtbaren Fall und den vollständig beobachtbaren Fall sind in den Abbildungen 4.9 und 4.10 dargestellt. Auffällig ist dabei die Verschlechterung des $\mathcal{Q}_1$-Netzes, die ausschließlich im Fall der partiellen Beobachtbarkeit und dort für das $\mathcal{Q}_1$-Netz auftritt. Die Ursachen hierfür stellen sich wie folgt dar. Das Modell $\mathcal{Q}_1$ approximiert die Q-Werte für Paare $(\tilde{\mathbf{s}}_1, a_1)$ aus rekonstruierten Zuständen $\tilde{\mathbf{s}}_1$ und Aktionen $a_1$ in Zeitschritt 1. Der rekonstruierte Zustand $\tilde{\mathbf{s}}_1$ wiederum setzt sich zusammen aus $a_0$ und $\mathbf{o}_1$ (vgl. Abschnitt 4.2). In späteren Zeitschritten stehen weitere Observablen $\mathbf{o}_t$ zur Verfügung die auf die aktuellen Prozessbedingungen schließen lassen. Eine Analyse der Daten zeigt, dass der Informationsgehalt der beobachtbaren Größen $\mathbf{o}_1$ bezüglich des Reibungskoeffizienten sehr gering ist und $\mathcal{Q}_1$ sich im Wesentlichen auf die Aktionswerte $a_0$ und $a_1$ verlässt. Während der frühen Episoden führt eine hohe Varianz in den Q-Funktionsapproximationen und die hohe Explorationsrate dazu, dass die Aktionen $a_0$ und $a_1$ annähernd gleichverteilt im *Replay Memory* vorliegen. In späteren Episoden sind die Entscheidungen der Lernstrategie in Zeitschritt 0 und 1 zunehmend stabiler. Die Aktionen $a_0$ und aufgrund des niedrigen Informationsgehalts von $\mathbf{o}_1$ auch $a_1$ werden unabhängig von den Prozessbedingungen gewählt. Die durch das Modell erklärbare Varianz in den Daten nimmt folglich ab, wodurch der Wert des Bestimmtheitsmaßes sinkt. Hingegen ist im Fall vollständiger Beobachtbarkeit (Abbildung 4.10) der Reibungskoeffizient explizit als Teil von $\mathbf{o}_0$ gegeben und ein optimales Verhalten in Abhängigkeit von dem Reibungskoeffizienten kann auch für $a_0$ und $a_1$ gelernt werden. Der Wert des Bestimmtheitsmaßes über die Zeit nimmt dann folglich auch für $\mathcal{Q}_1$ zu.

Ergebnisse von Untersuchungen der Dateneffizienz des BFHNQ Algorithmus im Fall gleichbleibender Prozessbedingungen werden in 4.4.2 vorgestellt. Den dargestellten Ergebnissen ist zu entnehmen, dass BFHNQ deutlich weniger Abtastungen des untersuchten Prozesses benötigt, um den optimalen Prozesspfad beziehungsweise einen nahezu optimalen Prozesspfad zu ermitteln als

der Hill-Climbing Ansatz. Dies begründet sich wie folgt. Während das vergleichsweise einfache Hill-Climbing Verfahren zufällig neu initialisiert wird, nachdem ein lokales Optimum erreicht wurde, wodurch jegliche Information über das Optimierungsproblem verworfen wird, lernt BFHNQ in Form der Q-Funktionen implizit eine Approximation der erwarteten Belohnungen und kann dieses Wissen zur gezielten Optimierung nutzen. Das Optimierungsverhalten der beiden Algorithmen ist in Anhang B.0.1 dargestellt.
Ergebnisse der Untersuchung der multikriteriellen Erweiterung des BFHNQ Algorithmus sind in 4.5.2 dargestellt. Hierbei wird dargestellt, dass der Algorithmus dank der Erweiterung in der Lage ist, gelerntes Prozesswissen auf sich ändernde Zielvorgaben zu übertragen und so dateneffizient unter neuen Zielvorgaben zu lernen. Gegen Ende der 1000 Episoden übertreffen die Ergebnisse des skalaren BFHNQ allerdings die Ergebnisse der multikriteriellen Erweiterung. Dies ist auf das nicht-Zutreffen der Annahme 4.15 zurückzuführen und es ist zu erwarten, dass sich diese Differenz der Ergebnisse für eine zunehmende Streuung der Zustandsübergänge weiter erhöht. Insbesondere auch das relativ gesehen schlechtere Abschneiden des SARSA Updates (siehe Abbildung 4.16) entspricht nicht den Erwartungen. Eine daraufhin durchgeführte Untersuchung des Ansatzes führt zu dem Schluss, dass ein Grund hierfür ist, dass die Verteilung der Zustands-Aktions Tupel $(s,a)$ in $\mathcal{D}$ sehr stark von der Verteilung abweicht die, durch die aktuelle Strategie erzeugt werden würde. Dieses Problem verschärft sich durch die Übertragung der Erfahrungs-Daten aus alten Konfigurationen, weswegen in Stufe $d$ eine Verschlechterung der Performanz auftritt. Ein möglicher Ansatz zur Behebung der in diesem Kapitel geschilderten, durch die nicht-lineare Skalarisierungsfunktion und die Nutzung historischer Daten hervorgerufenen Probleme wird in Kapitel 6 geschildert.

# 5 Struktur-geleitete Optimierung von Fertigungsprozessen

Im vorangegangenen Kapitel wird die Optimierung von Regelungsstrategien in Bezug auf die Bewertung des Prozessresultats anhand seiner Eigenschaften behandelt. Fertigungsprozesse wirken häufig direkt auf die Mikrostruktur des Werkstückes und beeinflussen so die Eigenschaften und die Leistungsfähigkeit des Materials. Diese Kausalitätskette Prozessausführung → Material-Struktur → Materialeigenschaften → Leistungsfähigkeit des Materials bildet das *three-link chain model* nach Olson [98] (Abbildung 5.1). Durch häufig nichtlineare komplexe Zusammenhänge zwischen Material-Struktur und -Eigenschaften ist es für die Prozessoptimierung vorteilhaft die Gütefunktion in dem Raum zu definieren in dem der Prozess wirkt, dem Raum der Material-Strukturen.

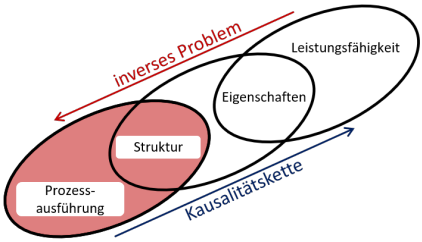



Abbildung 5.1: *three-link chain model* nach Olson [98], in abgewandelter Form. Rot markiert: der Fokus der Struktur-geleiteten Optimierung von Fertingsprozessen.

In diesem Teil der Arbeit werden deshalb Methoden des bestärkenden Lernens entwickelt und untersucht, die zum Ziel haben Prozesspfade von Fertigungsprozessen, welche die Struktur von Materialien verändern, zu optimieren. Die Optimierung hat das Erreichen einer bestimmten Zielstruktur oder eines Elements aus einer Menge von Zielstrukturen zum Ziel und wird durch die Kenntnis der aktuell vorliegenden Struktur geleitet. Zielstrukturen sind Strukturen, die vorgegebene Materialeigenschaften aufweisen. Die entwickelten Methoden sind für Strukturbeschreibungen auf beliebigen Skalen anwendbar. Im Kontext des *Material Design*, sowie in der unten vorgestellten Anwendungsstudie ist die Mikrostruktur von besonderem Interesse.

Übergreifendes Ziel bei der inversen Optimierung der oben beschriebenen Kausalitätskette ist es, eine Möglichkeit zu schaffen, für gewünschte Materialeigenschaften neue Materialien und zugehörige Fertigungspfade gezielt und automatisiert ermitteln zu können. Die Prozesspfadoptimierung mit Hinsicht auf damit erreichte Zielstrukturen löst die Problemstellung, welche sich aus dem letzten Glied der invertierten Kausalitätskette ergibt und ergänzt die Methoden zur Abbildung von Materialeigenschaften auf Material-Strukturen. Methoden zur Abbildung von Materialeigenschaften auf hierfür erforderliche Material-Strukturen basieren meist auf Methoden der Optimierung sowie des *überwachten Maschinellen Lernens*. Ein Auszug der in den letzten Jahren vorgestellten Methoden wird in 2.4.4 vorgestellt.

Wie in 2.4.4 gezeigt, nutzen die meisten verwandten Arbeiten zur Ermittlung von Prozesspfaden zur Erreichung von Zielstrukturen entweder vorberechnete Datenbanken oder davon abgeleitete generalisierende Modelle als Basis für eine Prozesspfad-Suche. Des Weiteren ist ein Großteil der vorgestellten Methoden nicht ohne Weiteres auf lange Prozesspfade anwendbar. Diese Beschränkung auf eine vorgegebene Lösungsmenge wird durch die in diesem Kapitel entwickelten Methoden des bestärkenden Lernens vermieden, da diese die Prozesspfade in direkter Interaktion mit dem Prozess beziehungsweise der Prozesssimulation optimieren. Außerdem sind die vorgestellten Methoden in der Lage auch sehr lange Prozesspfade zu optimieren.

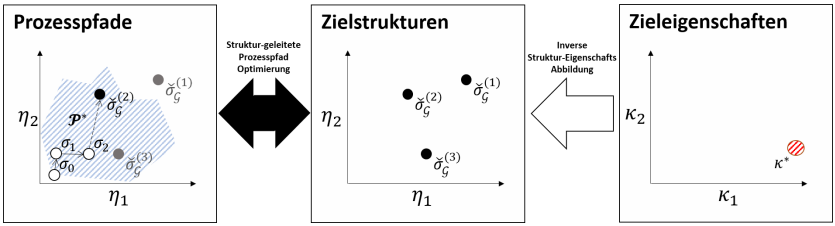



Abbildung 5.2: Einbettung der vorgestellten Methode im Kontext der inversen Optimierung der Kausalitätskette. Eine Methode zur inversen Struktur-Eigenschafts-Abbildung ermittelt Zieleigenschaften $\kappa^*$ aufweisende Strukturen $\breve{\sigma}_{\mathcal{G}}^{(g)}$. Durch Struktur-geleitete Prozesspfad Optimierung werden Prozesspfade ermittelt die von einer Anfangsstruktur $\sigma_0$ zu einer der Zielstrukturen führen.

Üblicherweise existieren mehrere Strukturen, die hinsichtlich der gewünschten Zieleigenschaften äquivalent sind. Aufgabe der Prozesspfadoptimierung ist dann, die am besten erreichbare Zielstruktur aus der Menge dieser äquivalenten Strukturen zu ermitteln und einen Prozesspfad für diese Zielstruktur zu optimieren. In Abbildung 5.2 ist für den Fall mehrerer äquivalenter Zielstrukturen die Einbettung der vorgestellten Methoden in die inverse Optimierung der Kausalitätskette Prozessausführung $\rightarrow$ Material-Struktur $\rightarrow$ Materialeigenschaften skizziert. Zieleigenschaften $\kappa^* \in \mathbb{R}^n$ liegen in einem Zielbereich des Eigenschaftsraums (rot schraffiert), der hier stellvertretend als zweidimensionaler Vektorraum mit den Komponenten $(\kappa_1, \kappa_2)$ skizziert ist. Durch Methoden zur inversen Struktur-Eigenschafts-Abbildung werden Material-Strukturen $\breve{\sigma}_{\mathcal{G}}^{(g)}$ ermittelt, welche die Zieleigenschaften aufweisen. Strukturen $\sigma$ werden durch eine numerische Strukturbeschreibung $\eta(\sigma) \in \mathbb{R}^m$ repräsentiert. Der Strukturbeschreibungsraum ist in Abbildung 5.2 stellvertretend in zweidimensionaler Form $(\eta_1, \eta_2)$ repräsentiert, liegt in der Anwendung aber meist in hoch-dimensionaler Form vor. Die ermittelten Strukturen $\breve{\sigma}_{\mathcal{G}}^{(g)}$ stellen Zielstrukturen dar, zu deren Erreichung im nächsten Schritt ein Prozesspfad gefunden werden muss. Dies ist die Aufgabe der *Struktur-geleiteten Prozesspfadoptimierung*: Der Suche nach einem Prozesspfad $\mathcal{P}^*$, der von einer initialen Struktur $\sigma_0$ zu

einer der äquivalenten Strukturen $\check{\sigma}_{\mathcal{G}}^{(g)}$ führt. Üblicherweise ist dabei durch Beschränkungen des Prozesses und des Materials nur ein Teilbereich des gesamten Strukturraums erreichbar (blau schraffierter Bereich).
Das Kapitel gliedert sich wie folgt: In Abschnitt 5.1 wird die oben beschriebene Aufgabenstellung formal eingeführt. In 5.2.1 wird ein Ansatz des tiefen bestärkenden Lernens zur Prozesspfadoptimierung für eine gegebene Zielstruktur entwickelt. In 5.2.2 wird ein erweiterter Ansatz zur effizienten Prozesspfadoptimierung für mehrere äquivalente Zielstrukturen entwickelt. Zur Analyse und Evaluation der entwickelten Methoden werden diese auf einen simulierten Metallverarbeitungsprozess angewandt, in welchem Zielstrukturen durch eine wählbare Folge aus einachsigen Druck- und Zug-Belastungen auf das Material in unterschiedliche Richtungen erreicht werden. Die Strukturbeschreibung liegt dabei in Form der kristallographischen Textur vor. Die Simulation dieses Prozesses wird in 5.3.1 vorgestellt. In 5.3.2 wird der Markov-Entscheidungsprozess für den Beispielprozess spezifiziert. Die entwickelten Methoden setzen die Definition einer Distanzfunktion im Strukturraum voraus. Für den Fall der Strukturbeschreibung durch die kristallographische Textur wird eine solche in 5.3.3 für kristallographische Texturen entwickelt. Die hier vorgestellten Methoden und Untersuchungen wurden als Zwischenergebnisse der Arbeit in [8] veröffentlicht.

## 5.1 Aufgabenstellung

Ziel der Struktur-geleiteten Optimierung ist die Optimierung von Prozesspfaden $\mathcal{P} = (a_0, a_1, .., a_K)$ der Länge $K \leq T$, ausgehend von einer initialen Struktur $\sigma_0$, wobei $T$ die maximal erlaubte Anzahl der Prozessschritte ist. Gegeben ist eine Zielstruktur $\check{\sigma} \in \Sigma$ oder eine Menge $\mathcal{G}$ von Zielstrukturen $\check{\sigma}_{\mathcal{G}}^{(g)} \in \mathcal{G}$. Die Erreichbarkeit der Zielstrukturen $\check{\sigma} \in \Sigma$, bzw. $\check{\sigma}_{\mathcal{G}}^{(g)}$ durch den Prozess ist nicht garantiert.

Für den Fall, dass eine einzelne Zielstruktur $\check{\sigma} \in \Sigma$ vorliegt, führt ein optimaler Prozesspfad $\mathcal{P}^*$ von der initialen Struktur $\sigma_0$ zu einer Struktur $\sigma^*$. Die Struktur $\sigma^*$ ist die der Zielstruktur $\check{\sigma} \in \Sigma$ bezüglich einer Struktur-Distanzfunktion $d_\sigma : \Sigma \times \Sigma \to \mathbb{R}_0^+$ am nächsten gelegene Struktur

$$\sigma^* = \underset{\sigma \in \Sigma_\mathcal{P}}{\arg\min} \left[ d_\sigma(\sigma, \check{\sigma}) \right], \tag{5.1}$$

aus der Menge der durch den Prozess erreichbaren Strukturen $\Sigma_\mathcal{P} \subseteq \Sigma$. Weder $\Sigma_\mathcal{P}$, noch $\sigma^*$ sind im Vorfeld der Optimierung bekannt.
Da die Struktur-Eigenschaftsabbildung üblicherweise eine $n:1$ Abbildung darstellt, sind häufig mehrere äquivalente Zielstrukturen $\check{\sigma}_\mathcal{G}^{(g)} \in \mathcal{G}$ anstelle der einzelnen Zielstruktur $\check{\sigma}$ gegeben. Ziel neben der Identifikation von $\sigma^*$ und des Pfades zu $\sigma^*$ ist dann die Identifikation der am besten erreichbaren Zielstruktur $\check{\sigma}_\mathcal{G}^* \in \mathcal{G}$:

$$(\sigma^*, \check{\sigma}_\mathcal{G}^*) = \underset{\left(\sigma, \check{\sigma}_\mathcal{G}^{(g)}\right) \in \Sigma_\mathcal{P} \times \mathcal{G}}{\arg\min} \left[ d_\sigma\left(\sigma, \check{\sigma}_\mathcal{G}^{(g)}\right) \right]. \tag{5.2}$$

Die beschriebenen Aufgabenstellungen können als Markov-Entscheidungsprozesse (MDP) mit endlichem Zeithorizont formalisiert werden. In 5.1.1 wird die Formulierung des Optimierungsproblems mit einer einzelnen Zielstruktur als MDP eingeführt. Darauf aufbauend wird in 5.1.2 ein neuartiger erweiterter MDP zur Behandlung von Fällen mit mehreren äquivalenten Zielstrukturen vorgestellt.

### 5.1.1 Markov-Entscheidungsprozess mit einer Zielstruktur

Für die Formalisierung als Markov-Entscheidungsprozess mit endlichem Zeithorizont $(S, A, P, R, \gamma, P_0, \bar{S})$ (siehe Kapitel 2.2.1) wird der Struktur-verändernde Prozess beschrieben durch eine Menge von Prozessaktionen $a_t \in A$ und verhält

sich folgend einer Zustandsübergangsfunktion $P$. Der Prozess befindet sich initial in Zustand $s_0$. Eine Prozessausführung endet nach maximal $T$ Zeitschritten. Ziel ist die Maximierung der erwarteten $\gamma$-diskontierten Belohnungen, gegeben durch die Belohnungsfunktion $R$.
Die Zustandsbeschreibung zum Zeitschritt $t$, $\mathbf{s}_t \in \mathbb{R}^{m+n}$ setzt sich zusammen aus einer numerischen, vektoriellen Repräsentation der aktuellen Material-Struktur $\sigma_t$, bezeichnet als $\eta(\sigma_t) \in \mathbb{R}^m$, für die angenommen wird, dass sie $\sigma_t$ bezüglich des Optimierungsziels vollständig charakterisiert. Gegebenenfalls wird die Zustandsbeschreibung um $n$ weitere Größen ergänzt, die zusätzliche Information über den Zustand beinhalten, um den Lernprozess zu erleichtern. Dies ist beispielsweise der aktuelle Zeitschritt.
Die Belohnungsfunktion $R$ wird basierend auf einer Struktur-Distanzfunktion $d_\sigma$ so gewählt, dass die Maximierung der Belohnung zu der in (5.1) eingeführten Struktur $\sigma^*$ führt:

$$R(s_t, a_t, s_{t+1}) = \begin{cases} \dfrac{1}{d_\sigma(\sigma_K, \check{\sigma})} & \text{, if } t = K-1, \\ 0 & \text{, else.} \end{cases} \tag{5.3}$$

Die so gewählte Belohnungsfunktion ist maximal für $\sigma_K = \sigma^*$. Sie hat außerdem die Eigenschaft, dass, für einen neutralen Diskontierungsfaktor ($\gamma = 1$), die für $\pi$ erwartete Belohnung $V_\pi(s)$ der erwarteten inversen Distanz der Struktur am Ende der jeweiligen Episode $\sigma_K$ zur Zielstruktur $\check{\sigma}$ entspricht. Im Fall eines deterministischen Prozesses ist die Bewertungsfunktion $V_\pi(s)$ maximal für eine Strategie $\pi$ die einen Prozesspfad zu $\sigma_K = \sigma^*$ erzeugt.

### 5.1.2 Markov-Entscheidungsprozess mit mehreren äquivalenten Zielen

Wie im vorangegangenen Unterabschnitt gezeigt, ist das Optimierungsproblem mit einer einzelnen Zielstruktur als Markov-Entscheidungsprozess mit endlichem Zeithorizont formulierbar. Im Fall mehrerer äquivalenter Zielstrukturen wäre dies durch eine Erweiterung der Belohnungsfunktion aus (5.3) möglich, indem am Episodenende das Belohnungssignal für die nächste der äquivalenten Zielstrukturen ausgegeben wird

$$R(s_t, a_t, s_{t+1}) = \begin{cases} \dfrac{1}{\arg\min_{\breve{\sigma}_{\mathcal{G}}^{(g)} \in \mathcal{G}} \left( d_\sigma(\sigma_K, \breve{\sigma}_{\mathcal{G}}^{(g)}) \right)} & \text{, if } t = K-1, \\ 0 & \text{, else.} \end{cases} \tag{5.4}$$

Diese Erweiterung der Belohnungsfunktion bringt allerdings Nachteile mit sich. Im Gegensatz zu der Formulierung für einzelne Zielstrukturen existiert nun pro Zielstruktur ein lokales Maximum der Belohnungsfunktion bezüglich $\sigma_K$. Dies erschwert das Lernen und kann dazu führen, dass der Agent sich, trotz Exploration, beim Lernen früh auf eine sub-optimale Zielstruktur festlegt. Außerdem erschwert eine solche Erweiterung die Umformung der Belohnungsfunktion, die in 5.2.1 als Teil der Lösung der Struktur-geleiteten Optimierung vorgestellt wird und, wie in 5.4.1 gezeigt, einen wesentlichen Einfluss auf die Konvergenzgeschwindigkeit der entwickelten Methoden hat.

Die Umformulierung der Belohnungsfunktion und die damit verbundenen Nachteile lassen sich vermeiden, indem der Agent während des Lernens die verfolgte Zielstruktur explizit festlegt und die Belohnungsfunktion in Abhängigkeit dieser verfolgten Zielstruktur definiert wird. Zu diesem Zweck wird als

Basis für die vorgestellten Methoden zur Optimierung mit mehreren äquivalenten Zielstrukturen eine neuartige erweiterte Form von Markov-Entscheidungsprozessen mit endlichem Zeithorizont eingeführt: *Markov-Entscheidungsprozesse mit mehreren äquivalenten Zielen* (*multi-equivalent Goal MDPs*, MEG-MDP.).

Ein MEG-MDP ist eine erweiterte Form des in 2.2.1 vorgestellten MDPs mit endlichem Zeithorizont. Er ist definiert als 8-Tupel $(S, s_0, A, P, \gamma, \bar{S}, R_g, G)$. Die Definition der Prozessdynamik durch die Zustandsübergangsfunktion $P$, definiert über dem Zustandsraum $S$ und dem Aktionsraum $A$, sowie der Einfluss des Diskontierungsfaktors $\gamma$ bleibt dabei unverändert gegenüber dem MDP mit endlichem Zeithorizont. Der Zustand $s_0$ wird als gleichbleibender Anfangszustand angenommen und $\bar{S}$ ist die Menge der möglichen Endzustände. Zusätzlich ist eine Menge äquivalenter Ziele $g \in G$ gegeben und die Belohnungsfunktion $R_g(s, a, s')$ in Abhängigkeit des Ziels $g$ definiert. Pro Ziel $g \in G$ existiert eine optimale Strategie $\pi_g^*$. Da Ziele $g \in G$ äquivalent bezüglich des Optimierungsziels sind, ist die optimale zielübergreifende Strategie als Lösung des MEG-MDPs definiert als $\pi^* = \pi_{g^*}^*$ für $g^* = \arg\max_{g \in G} V_g^*(s_0)$, wobei $V_g^*(s_0)$ der Bewertung des Anfangszustands $s_0$ bezüglich der Strategie $\pi_g^*$ und der Belohnungsfunktion $R_g$ entspricht.

Im Fall der Struktur-geleiteten Optimierung entspricht ein Ziel $g \in G$ einer Zielstruktur $\breve{\sigma}_{\mathcal{G}}^{(g)} \in \mathcal{G}$. Analog zu der zielabhängigen optimalen Strategie $\pi_g^*$ existiert ein optimaler Prozesspfad $\mathcal{P}_g^*$ pro Zielstruktur $\breve{\sigma}_{\mathcal{G}}^{(g)}$. Der optimale zielübergreifende Prozesspfad $\mathcal{P}^*$ entspricht der Realisierung von $\pi^*$ ausgehend von $s_0$.

## 5.2 Lösungsmethoden

Die dargestellte Aufgabenstellung unterscheidet sich in einigen Punkten von der Aufgabenstellung bei der Optimierung partiell beobachtbarer Fertigungsprozesse unter stochastischen Einflüssen (Kapitel 4). Aus diesem Grund wurden im Rahmen der Arbeit spezielle Lösungsmethoden entwickelt, die im Kern aber auf demselben Lösungsansatz beruhen: Dem modellfreien bestärkenden Lernen auf Basis von Bewertungsfunktionen. Wesentliche Unterscheidungsmerkmale der Aufgabenstellungen sind im Einzelnen:

1. Während in Kapitel 4 eine effiziente Lösungsmethode für die Optimierung von Fertigungsprozessen mit kurzem Zeithorizont vorgestellt wurde, sollen die hier vorgestellten Methoden auch auf Optimierungsprobleme mit langem Zeithorizont anwendbar sein.

2. Anders als in Kapitel 4, wo der Prozesszustand nur partiell beobachtbar war, wird hier von einer vollständigen Beobachtbarkeit und der Repräsentation der Material-Struktur in einem hochdimensionalen Strukturbeschreibungsraum ausgegangen.

3. Der in 5.1.2 eingeführte MEG-MDP motiviert sich aus der Tatsache, dass bei der inversen Optimierung der Kausalitätskette Prozess, Struktur, Eigenschaft zu gewünschten Materialeigenschaften eine Menge resultierender Zielstrukturen ermittelt wird. Etwas vergleichbares wurde in Kapitel 4 nicht behandelt.

Der in Kapitel 4 vorgestellte Algorithmus *BackwardFHNQ* approximiert die Q-Funktion pro Zeitschritt $t \in [0,..,T]$ jeweils durch ein separates Modell $\mathcal{Q}_t$ (siehe 4.2). Dies hat den Vorteil, dass die Dimension der Zustandsbeschreibung $\mathbf{s_t}$ in Abhängigkeit des Zeitschritts variieren kann, was zur Optimierung des partiell beobachtbaren Prozesses genutzt wird. Außerdem ermöglicht es das effiziente Training der Erwartungswertmodelle, die bei *BackwardFHNQ* regelmäßig

von Grund auf neu trainiert werden. Die Zeitschritt-abhängige separate Approximation der Q-Funktion hat allerdings den Nachteil, dass eine Zeitschritt-übergreifende Generalisierung der approximierten Bewertungen ausgeschlossen ist. Außerdem erhöht sich der Trainingsaufwand bei *BackwardFHNQ* mit zunehmend hohem Zeithorizont $T$.

Im Gegensatz dazu wird bei den in 5.1.1 und 5.1.2 eingeführten Markov-Entscheidungsprozessen zur Struktur-geleiteten Optimierung davon ausgegangen, dass die Beschreibung $\eta(\sigma_t)$ der aktuellen Material-Struktur $\sigma_t$ in jedem Zeitschritt bekannt ist und $\mathbf{s}_t$ bezüglich des Optimierungsproblems durch $\eta(\sigma_t)$ vollständig charakterisiert ist[1]. Die Zustandsbeschreibung $\mathbf{s_t}$ besteht im Wesentlichen aus der vektoriellen Strukturbeschreibung $\eta(\sigma_t) \in \mathbb{R}^m$ und hält sich in einem Zeitschritt-unabhängigen Vektorraum $\mathbb{R}^{m+n}$ auf (mit $n$ zusätzlichen Variablen zur Vereinfachung des Lernproblems, siehe 5.1.1). Des Weiteren sind Beschreibungen der Material-Struktur $\eta(\sigma)$ in vielen Fällen von hoher Dimension $m$ und es sollen auch Fälle mit weitem Zeithorizont $T$ berücksichtigt werden (in dem untersuchten Anwendungsfall ist $m = 42$, $n = 2$ und $T = 100$, siehe Abschnitt 5.3).

Aus diesen Erwägungen heraus, die sich aus den Unterscheidungsmerkmalen 1. und 2. ergeben, werden zur Lösung der in Abschnitt 5.1 eingeführten Aufgabenstellung Methoden entwickelt, bei denen Bewertungsfunktion Zeitschritt-unabhängig durch ein einzelnes Modell approximiert wird. Dies wirkt sich darin aus, dass die zu approximierende Funktion, verglichen mit den Zeitschritt-abhängigen Funktionen, von höherer Komplexität ist. Gleichzeitig stehen allerdings mehr Daten zum Training des einzelnen Modells zur Verfügung. Das Training rückwärts in den Zeitschritten, durch Nutzung des bereits trainierten Modells $\mathcal{Q}_{(t+1)}$ (siehe 4.2) entfällt. Hinzu kommt, dass insbesondere der Zustandsraum von vergleichsweise hoher Dimension ist. Diese Faktoren führen

[1] in Abschnitt 6.2 wird eine Erweiterungsmöglichkeit der in diesem Kapitel vorgestellten Methoden für den Fall der partiellen Beobachtbarkeit skizziert.

im direkten Vergleich mit Kapitel 4 dazu, dass hier Modelle mit deutlich höherer Kapazität zur Approximation genutzt werden müssen, wodurch der Trainingsaufwand bedeutend steigt. Anstelle des wiederholten neu-Trainings der $\mathcal{Q}$-Netze ist hier deshalb ein Algorithmus mit fortlaufend verfeinernd trainierten $\mathcal{Q}$-Netzen Voraussetzung für effizientes Lernen.
Unterscheidungsmerkmal 3. bezieht sich auf die Besonderheit der äquivalenten Ziele bei der Formulierung der Prozessoptimierung im Raum der Material-Strukturen. Algorithmen können zwei grundsätzlich unterscheidbare Ansätze verfolgen, um damit umzugehen:

(a) In einem ersten Schritt werden optimale Prozesspfade $\mathcal{P}_g^*$ pro $\check{\sigma}_\mathcal{G} \in \mathcal{G}$ ermittelt, um retrospektiv den optimalen Prozesspfad $\mathcal{P}^* = \arg\max_{g \in G} \mathcal{P}_g^*$ und damit das optimale Ziel $\check{\sigma}_\mathcal{G}^*$ zu identifizieren.

(b) Die Identifikation der am besten erreichbaren Zielstruktur $\check{\sigma}_\mathcal{G}^*$ wird als integraler Bestandteil der Optimierung betrachtet und simultan mit der Optimierung der Prozesspfade gelöst.

Für ein beliebiges Optimierungsverfahren für Probleme mit einzelnen Zielstrukturen ist Ansatz (a) auf triviale Art umsetzbar indem das Verfahren auf jede Zielstruktur $\check{\sigma}_\mathcal{G}$ separat angewandt wird. Diese separate Betrachtung bedeutet allerdings, dass für die Lösung des übergreifenden Optimierungsproblems das Verfahren für jede Zielstruktur in $\mathcal{G}$ bis zur Konvergenz angewandt werden muss und die Kosten der Optimierung von der Kardinalität der Menge $\mathcal{G}$ abhängen. Wenn die Optimierung modellfrei in direkter Interaktion mit dem Prozess beziehungsweise einer Simulation des Prozesses stattfindet, entsprechen diese Kosten der Anzahl der zur Optimierung notwendigen Interaktionen. Im Fall der Interaktion mit einer Simulation schlägt sich dies direkt im zeitlich/rechnerischen Simulationsaufwand nieder. Im Fall der direkten Interaktion mit einem Realprozess spielen zusätzliche Größen wie Werkzeugverschleiß und Materialverbrauch während der Optimierung eine Rolle.
Im Unterschied dazu ist es bei der simultanen Identifikation von $\check{\sigma}_\mathcal{G}^*$ und Prozesspfadoptimierung möglich, die genannten Ressourcen schon frühzeitig für

Zielstrukturen zu nutzen, die mit hoher Wahrscheinlichkeit effizient erreichbar sind und Zielstrukturen zu ignorieren, die als unerreichbar angesehen werden. Daher wird in 5.2.2 ein Ansatz des bestärkenden Lernens vorgestellt, der bei der Lösung von Struktur-geleiteten Optimierungsproblemen Ansatz (b) verfolgt.
Die Wahl des Basisalgorithmus des bestärkenden Lernens wurde aufgrund der hier spezifizierten Anforderungen getroffen. Die in 5.2.1 entwickelten Methoden zur effizienten Struktur-geleiteten Optimierung mit einzelnen Zielstrukturen und die in 5.2.2 entwickelte Methode für mehrere äquivalente Ziele sind jedoch von generischer Natur und können in verschiedene Basisalgorithmen des bestärkenden Lernens integriert werden.

### 5.2.1 Struktur-geleitete Optimierung mit einzelnen Zielstrukturen

Die Struktur-geleitete Optimierung mit einzelnen Zielstrukturen ist als Problem des bestärkenden Lernens zusammenfassend in Abbildung 5.3 illustriert. Optimiert wird ein Prozesspfad $\mathcal{P}$, der im Strukturraum Strukturen $[\sigma_0, \sigma_1, ..., \sigma_{t+1}]$ miteinander verbindet. Der Strukturraum ist abstrahiert als zwei-dimensionaler Vektorraum dargestellt. Die Menge der durch den Prozess erreichbaren Strukturen liegt in dem schraffiert dargestellten Bereich. Aktionen $a_t$ des Prozesspfades werden durch den Agenten vorgegeben. Der Agent wird geleitet durch Strukturbeschreibungen $\mathbf{s}_t = \eta(\sigma_t)$ und ein auf der Strukturdistanz $d_\sigma(\sigma_t, \check{\sigma})$ beruhendes Belohnungssignal. Ziel der Optimierung ist die Ermittlung eines Prozesspfades von $\sigma_0$ zu $\sigma^*$ (5.1). Das durch die Formulierung als Markov-Entscheidungsprozess tatsächlich verfolgte Ziel ist die Maximierung der erwarteten Belohnung (5.3). Beide Ziele sind für $\gamma = 1$ in Übereinstimmung (siehe 5.1.1).
Bei Formulierung der Belohnungsfunktion aus (5.3) tritt ein Belohnungssignal ungleich Null nur am Ende einer Episode beim Übergang von $s_{K-1}$ zu $s_K$ auf.

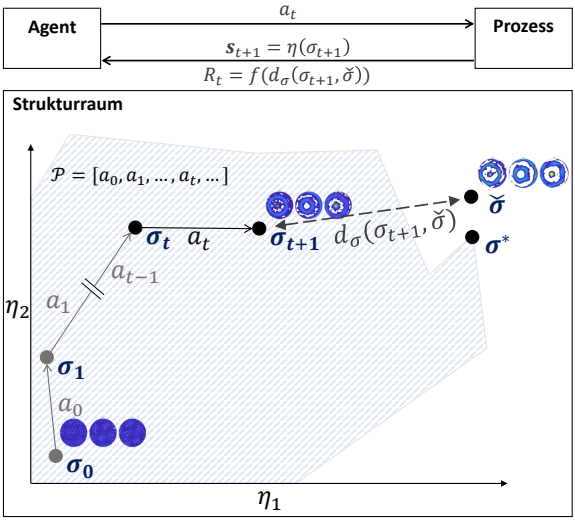



Abbildung 5.3: Aufgabenstellung der *Struktur-geleitete Optimierung mit einzelnen Zielstrukturen*. Ziel ist die Optimierung eines Prozesspfades $\mathcal{P}$ zur Erreichung der Zielstruktur $\hat{\sigma}$ von einer Anfangsstruktur $\sigma_0$ ausgehend. Der Optimierungs-Agent wird dabei geleitet von Beschreibungen der Struktur $\eta(\sigma)$ und von Belohnungssignalen $R$, die von der Distanz im Strukturraum $d_\sigma$ abhängen.
Dem unten beschriebenen Anwendungsfall vorgreifend, werden jeweils die $(001)$-,$(011)$-,$(111)$-Polfiguren zur Strukturvisualisierung

Insbesondere im Fall von langen Prozesspfaden (beziehungsweise Problemen mit hohem Zeithorizont $T$) kann dies dazu führen, dass sehr viele Samples benötigt werden, um den Einfluss einzelner Aktionen der Episode in Bezug auf das Belohnungssignal zu bestimmen[2]. Insbesondere in Fällen, bei denen die Prozessausführung ressourcenintensiv ist, kann dies in der Praxis dazu führen, dass bestärkendes Lernen ohne weitere Modifikation nicht mit vertretbaren Mengen von Ressourcen vereinbar ist. Eine Möglichkeit, diesem Problem zu begegnen, ist das *Potential-Based Reward Shaping*, eingeführt in [51]. Zum

[2] dem sogenannten *Credit Assignment Problem* [16]

Lernen wird hierbei die Belohnungsfunktion $R$ durch eine umgeformte Variante

$$R' = R + \mathcal{F}. \tag{5.5}$$

substituiert. Die Umformfunktion $\mathcal{F}$ ist dabei definiert als

$$\mathcal{F}(s_t, s_{t+1}) = \gamma\Phi(s_{t+1}) - \Phi(s_t), \tag{5.6}$$

wobei $\gamma$ der Discout-Faktor des jeweiligen MDPs ist und $\Phi : S \to \mathbb{R}$ eine Potentialfunktion. Wie in [51] gezeigt, ist bei Entscheidungsprozessen *mit endlichem Zeithorizont* mit einem einzelnen Endzustand die optimale Strategie $\pi^*$ und Strategien nahe $\pi^*$ invariant bezüglich der Substitution von $R$ durch $R'$. Für Entscheidungsprozesse mit mehreren Endzuständen ist die Invarianz garantiert, wenn für den Endzustand $s_K$ jeder Episode $\Phi(s_K) = 0$ gilt (siehe [99]).
Um dieser Anforderung gerecht zu werden, wird in dieser Arbeit folgende Potentialfunktion definiert:

$$\Phi(s_t) = \begin{cases} 0 & \text{, if } t = K \\ \dfrac{1}{d_\sigma(\sigma_t, \breve{\sigma})} & \text{, else.} \end{cases} \tag{5.7}$$

Nach der Substitution von (5.3), (5.6) und (5.7) in die Umformungsgleichung (5.5) ist $R'(s_t, a_t, s_{t+1})$ definiert durch

$$\begin{aligned} R'(s_t, a_t, s_{t+1}) &= \begin{cases} \dfrac{1}{d_\sigma(\sigma_K, \breve{\sigma})} + \gamma\Phi(s_K) - \Phi(s_t) & \text{, if } t = K-1 \\ \gamma\Phi(s_{t+1}) - \Phi(s_t) & \text{, else} \end{cases} \\ &= \begin{cases} \dfrac{1}{d_\sigma(\sigma_{t+1}, \breve{\sigma})} - \dfrac{1}{d_\sigma(\sigma_t, \breve{\sigma})} & \text{, if } t = K-1 \\ \dfrac{\gamma}{d_\sigma(\sigma_{t+1}, \breve{\sigma})} - \dfrac{1}{d_\sigma(\sigma_t, \breve{\sigma})} & \text{, else.} \end{cases} \end{aligned} \tag{5.8}$$

Für den Fall, dass der Discount-Faktor neutral ist ($\gamma = 1$) gilt in jedem Zeitschritt $R' = \frac{1}{d_\sigma(\sigma_{t+1}, \check{\sigma})} - \frac{1}{d_\sigma(\sigma_t, \check{\sigma})}$.
Im Gegensatz zu der Belohnungsfunktion $R$ aus (5.3), emittiert die umgeformte Variante $R'$ Belohnungssignale ungleich null auch während der Episode und ermöglicht so effizientes Lernen. Aufgrund der oben genannten Invarianzen kann $R'$ während des Lernens anstelle von $R$ genutzt werden, ohne das Optimierungsproblem zu verfälschen.
Die beschriebene Formulierung der Struktur-geleiteten Prozesspfadoptimierung als Markov-Entscheidungsprozess, die Umformung der Belohnungsfunktion, sowie die im folgenden Unterabschnitt beschriebenen Methoden zur effizienten Lösung von Problemen mit mehreren äquivalenten Zielstrukturen können in verschiedene Basisalgorithmen des bestärkenden Lernens integriert werden. Aus diesem Grund wird hier, wie bei der Optimierung partiell beobachtbarer Fertigungsprozesse in Kapitel 4 die Anwendung eines approximativen *Q-Learning* Algorithmus als Basisalgorithmus untersucht. Aufgrund der oben beschriebenen Charakteristika der Problemstellung und den davon abgeleiteten Anforderungen wird ein Basisalgorithmus gewählt, bei dem ein einzelnes Zeitschritt-unabhängiges $\mathcal{Q}$-Netz fortlaufend verfeinernd trainiert wird. Der in 2.2.7 vorgestellte *Deep Q Networks* (DQN) [23] Algorithmus erfüllt diese Anforderungen. Für eine erhöhte Dateneffizienz und Lern-Stabilität werden außerdem die in 2.2.7 vorgestellten Erweiterungen *Prioritized Experience Replay* [24], *Double Q-Learning* [25], und *Dueling Q-Learning* [26] genutzt. Als Lernstrategie wird die in 2.2.7 eingeführte $\varepsilon$-*greedy* Strategie genutzt. wobei [23] folgend, die initiale Explorationsrate $\varepsilon_0$ während der ersten $n_\varepsilon$ Episoden linear $\varepsilon_\mathrm{f}$ annähert und dann anschließend konstant $\varepsilon_\mathrm{f}$ beträgt.

### 5.2.2 Struktur-geleitete Optimierung mit mehreren äquivalenten Zielstrukturen

Ausgehend von der im vorangegangenen Unterabschnitt vorgestellten Lösungsmethode mit einzelnen Zielstrukturen und der Formulierung der MEG-MDPs in 5.1.2 wird hier eine erweiterte Methode zur effizienten Lösung von Optimierungsproblemen mit mehreren äquivalenten Zielstrukturen vorgestellt. Dieser wird im Folgenden als MEG-SGPPO Algorithmus (für *Multi-equivalent-Goal Structure-guided Processing Path Optimization*) bezeichnet, während der in 5.2.1 entwickelte grundlegende Algorithmus als SG-SGPPO (für *Single-Goal Structure-guided Processing Path Optimization*) bezeichnet wird. In Abbildung 5.4 ist die erweiterte Aufgabenstellung zusammenfassend skizziert. Anstelle einer einzelnen Zielstruktur $\breve{\sigma}$ ist nun eine Menge äquivalenter Zielstrukturen $\breve{\sigma}_{\mathcal{G}}^{(g)} \in \mathcal{G}$ gegeben. Die Definition des Optimierungsziels aus (5.1) erweitert sich dann zu (5.2) und die Optimierung des Prozesspfades ist mit der Identifikation der am besten erreichbaren Zielstruktur $\breve{\sigma}_{\mathcal{G}}^{*} \in \mathcal{G}$ verknüpft.

Der in 5.1.2 eingeführte MEG-MDP ist definiert durch das Tupel $(S, s_0, A, P, \gamma, T, R_g, G)$. Wie im vorangegangenen Unterabschnitt wird *Potential-Based Reward Shaping* genutzt um die nun Ziel-abhängige Belohnungsfunktion $R_g$ analog zu (5.8) umzuformen

$$R'_{\mathrm{g}} = \begin{cases} \dfrac{1}{d_\sigma\left(\sigma_{t+1}, \breve{\sigma}_{\mathcal{G}}^{(g)}\right)} - \dfrac{1}{d_\sigma\left(\sigma_t, \breve{\sigma}_{\mathcal{G}}^{(g)}\right)} & , \text{if } t = K-1 \\ \dfrac{\gamma}{d_\sigma\left(\sigma_{t+1}, \breve{\sigma}_{\mathcal{G}}^{(g)}\right)} - \dfrac{1}{d_\sigma\left(\sigma_t, \breve{\sigma}_{\mathcal{G}}^{(g)}\right)} & , \text{else.} \end{cases} \tag{5.9}$$

Für die Ziel-abhängige umgeformte Belohnungsfunktion $R'_{\mathrm{g}}$ werden Ziel-abhängige Bewertungsfunktionen gelernt. Die Zustandsbewertungsfunktion, wie sie in (2.3) in rekursiver Form eingeführt wurde, ist dann gegeben als

$$V'_{g,\pi}(s_t) = \mathbb{E}_{\pi,P}\left[R'_{\mathrm{g}}(s_t, a_t, s_{t+1}) + \gamma V'_{g,\pi}(s_{t+1})\right] \tag{5.10}$$

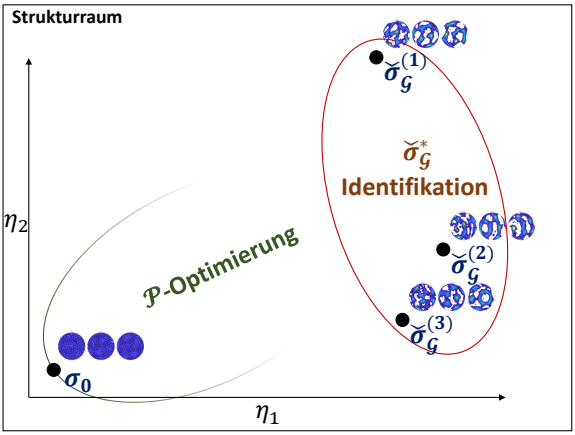



Abbildung 5.4: Aufgabenstellung der *Struktur-geleiteten Optimierung mit mehreren äquivalenten Zielstrukturen*. Gegeben ist eine Menge $\mathcal{G}$ bezüglich der Zieleigenschaften äquivalenter Zielstrukturen $\breve{\sigma}_{\mathcal{G}}^{(g)} \in \mathcal{G}$. Ziel ist die Identifikation der durch den Prozess am besten erreichbaren Struktur $\breve{\sigma}_{\mathcal{G}}^{*}$ und die gleichzeitige Optimierung eines Prozesspfades $\mathcal{P}$ zur Erreichung von $\breve{\sigma}_{\mathcal{G}}^{*}$.
Dem unten beschriebenen Anwendungsfall vorgreifend, werden jeweils die $(001)$-,$(011)$-,$(111)$-Polfiguren zur Strukturvisualisierung genutzt.

Die optimale Zustandsbewertungsfunktion für $g$ bezüglich $R'_g$ ist dann gegeben als $V'^{*}_g = V'_{g,\pi^*_g}$. Auf gleiche Art wird die Q-Funktion $Q_\pi$ durch die zielabhängige Q-Funktion $Q'_{g,\pi}$ bezüglich $R'_g$ und die in 2.2.7 eingeführte *Advantage*-Funktion $\bar{A}_\pi$ des *Dueling Q-Learning* durch die zielabhängige *Advantage*-Funktion $\bar{A}'_{g,\pi}$ bezüglich $R'_g$ ersetzt. Auch hierbei gilt $Q'^{*}_g = Q'_{g,\pi^*_g}$ und $\bar{A}'^{*}_g = \bar{A}'_{g,\pi^*_g}$.
Dem *universal function approximators* Ansatz aus [46] folgend werden zur Approximation der zielabhängigen Bewertungsfunktionen diese um eine numerische Repräsentation des Ziels als Parameter erweitert. Diese Repräsentation ist in dem vorliegenden Fall durch die Strukturbeschreibung $\eta\left(\breve{\sigma}_{\mathcal{G}}^{(g)}\right)$ gegeben. Hierdurch wird die Generalisierung der Bewertungsfunktionen über verschiedene Zielstrukturen hinweg ermöglicht. Die optimale zielabhängige Zustandsbewertungsfunktion $V'^{*}_g$ bezüglich $R'_g$ wird durch das Modell $\breve{\mathcal{V}}\left(\mathbf{s}, \eta\left(\breve{\sigma}_{\mathcal{G}}^{(g)}\right), \theta\right)$

approximativ gelernt. Auf gleiche Art werden $Q_g'^*$ durch $\check{\mathcal{Q}}\left(\mathbf{s}, a, \eta\left(\check{\sigma}_{\mathcal{G}}^{(g)}\right), \theta\right)$ und $A_g'^*$ durch $\check{\mathcal{A}}\left(\mathbf{s}, a, \eta\left(\check{\sigma}_{\mathcal{G}}^{(g)}\right), \theta\right)$ angenähert.
Der hier vorgestellte Algorithmus zum Lösen von MEG-MDPs basiert auf der Priorisierung der Zielstrukturen zu Beginn einer Episode und der approximativen Identifikation der am besten erreichbaren Zielstruktur $\check{\sigma}_{\mathcal{G}}^*$ anhand der approximierten Bewertungsfunktionen. Hierbei kommt die in 5.1.1 beschriebene Eigenschaft der Belohnungsfunktion aus (5.3) zum Tragen, dass für $\gamma = 1$ die für Strategie $\pi$ erwartete zukünftige Belohnung dem Erwartungswert der inversen Distanz der am Ende der Episode erreichten Struktur $\sigma_K$ zur Zielstruktur $\check{\sigma}$ entspricht. Für die optimale Ziel-abhängige Zustands-Bewertungsfunktion $V_g^*$ ist dann $\check{\sigma}_{\mathcal{G}}^*$ identifizierbar durch

$$\check{\sigma}_{\mathcal{G}}^* = \underset{g \in G}{\arg\max} \left[V_g(s_0)\right]. \tag{5.11}$$

Durch die beschriebene Umformung der Belohnungsfunktion werden anstelle von $V_g^*$ Erwartungswerte $V_g'^*$ bezüglich der umgeformten Belohnungsfunktion $R_g'^*$ approximiert. Wenn für die Potentialfunktion, wie im vorliegenden Fall (siehe (5.7)), $\Phi(s_K) = 0$ gilt, gilt für alle Zustände $s \in S$ auch

$$V_{\mathrm{g}}'^*(s) = V_{\mathrm{g}}^*(s) - \Phi(s) \tag{5.12}$$

(siehe [100]).
Für die in (5.7) definierte Potentialfunktion ist $V_g^*(s_0)$ aus $V_g'^*(s_0)$ durch Anwenden der Gleichung

$$V_g^*(s_0) = V_g'^*(s_0) + \frac{1}{d_\sigma(\sigma_0, \check{\sigma}_{\mathcal{G}}^{(g)})} \tag{5.13}$$

rekonstruierbar.

Ausgehend von der gelernten Approximation der optimalen Zustandsbewertungsfunktion $\check{\mathcal{V}} \approx V_g'^*$ bezüglich $R_g'$ und durch Substitution von (5.13) in (5.11) ist eine Schätzung der am besten erreichbaren Struktur $\check{\sigma}_{\mathcal{G}}^{\sim}$ gegeben durch

$$\check{\sigma}_{\mathcal{G}}^{\sim} = \underset{g \in G}{\arg\max} \left[ \check{\mathcal{V}}\left(s_0, \eta\left(\check{\sigma}_{\mathcal{G}}^{(g)}\right), \theta\right) + \frac{1}{d_\sigma\left(\sigma_0, \check{\sigma}_{\mathcal{G}}^{(g)}\right)} \right]. \tag{5.14}$$

Die Genauigkeit der Schätzung $\check{\sigma}_{\mathcal{G}}^{\sim} \approx \check{\sigma}_{\mathcal{G}}^{*}$ während des Lernens ist abhängig von der aktuellen Qualität der geschätzten Bewertungsfunktion $\check{\mathcal{V}}$. Um zu garantieren, dass $\check{\sigma}_{\mathcal{G}}^{*}$ oder ein Ziel nahe $\check{\sigma}_{\mathcal{G}}^{*}$ gefunden wird, muss die Wahl der durch den Agenten verfolgten Zielstruktur $\check{\sigma} \in \mathcal{G}$ während des Lernens variieren. Wie bei der Bestimmung der auszuführenden Aktion $a \in A$ ergibt sich hier eine grundlegende Abwägung zwischen der Ausnutzung der aktuellen Bewertungsfunktion (*Exploitation*) und der Exploration zur Erlangung neuen Wissens (siehe 2.2.5).

Wie bei der Definition der Lernstrategie $\tilde{\pi}$ (siehe 2.2.6) kann auch hier eine $\varepsilon$-*greedy* Auswahlstrategie genutzt werden, um sicher zu stellen, dass alle Zielstrukturen berücksichtigt werden (Exploration), der Agent sich gleichzeitig aber auch auf vielversprechende Zielstrukturen fokussiert (*Exploitation*). $\check{\sigma}$ wird hierzu zu Beginn jeder Episode neu zugewiesen

$$\check{\sigma} \leftarrow \begin{cases} \check{\sigma}_{\mathcal{G}}^{\sim} & \text{, if } x \geq \check{\varepsilon}, \text{ where } x \sim \mathcal{U}(0,1), \\ \sigma \sim \mathcal{U}_{\mathcal{G}} & \text{, else,} \end{cases} \tag{5.15}$$

wobei $\mathcal{U}_{\mathcal{G}}$ die Gleichverteilung über die Zielmenge $\mathcal{G}$ und $\mathcal{U}(0,1)$ die Gleichverteilung im Intervall $[0,1]$ darstellt. Die Explorationsrate der Zielauswahl $\check{\varepsilon}$ in Abhängigkeit von der aktuellen Episode $e$ wird dabei, analog zu $\varepsilon$, durch drei Parameter definiert. Die initiale Explorationsrate $\check{\varepsilon}_0$ nähert linear die finale Explorationsrate $\check{\varepsilon}_\mathrm{f}$ während der ersten $n_{\check{\varepsilon}}$ an.

Der gesamte Lösungsansatz für mehrere äquivalente Zielstrukturen ist in Listing 9 dargestellt. Zu Beginn jeder Episode wird die verfolgte Zielstruktur $\check{\sigma}$

MEG-SGPPO($n_e, \mathcal{G}, \check{\varepsilon}_0, \check{\varepsilon}_\mathrm{f}, n_{\check{\varepsilon}}, DQN parameters$)

1 $\mathcal{D} \leftarrow \emptyset$
2 **for** $e = 1$ **to** $n_e$
3 $\quad \check{\varepsilon} \leftarrow \max\left(\check{\varepsilon}_\mathrm{f}, \check{\varepsilon}_0 - \frac{e}{n_{\check{\varepsilon}}}(\check{\varepsilon}_0 - \check{\varepsilon}_\mathrm{f})\right)$
4 $\quad \check{\sigma} \leftarrow$ Eq. (5.15)
5 $\quad \mathcal{E}_e \leftarrow$ execute *multi-goal DQN* wrt. $\check{\sigma}$ for one episode
6 $\quad$ **for** $g \in G$
7 $\quad\quad \mathcal{E}_\mathrm{e}^{(\mathrm{g})} \leftarrow \left\{\left(s_t, a_t, s_{t+1}, R'_g, \eta\left(\check{\sigma}_\mathcal{G}^{(g)}\right)\right) \middle| \left(s_t, a_t, s_{t+1}, R\right) \in \mathcal{E}_e\right\}$, where $R'_g$ wrt. $g$ is given by (5.9)
8 $\quad\quad \mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{E}_\mathrm{e}^{(\mathrm{g})}$

Listing 9. Multi-equivalent-Goal Structure-guided Processing Path Optimization (MEG-SGPPO).

festgelegt (Zeilen 3 und 4). Zugunsten einer übersichtlichen Darstellung fassen wir die Ausführung des angepassten DQN Algorithmus pro Episode in einer Zeile zusammen (Zeile 5). Der angepasste DQN Algorithmus entspricht dabei DQN wie es in 2.2.7 vorgestellt wurde in Kombination mit Funktionsapproximationsmodellen die, wie oben beschrieben, über Zieltexturbeschreibungen $\eta\left(\check{\sigma}_\mathcal{G}^{(g)}\right)$ generalisieren. Außerdem werden Erfahrungs-Tupel durch den angepassten DQN Algorithmus nicht direkt in das *Replay Memory* $\mathcal{D}$ übernommen, sondern pro Episode als separate Menge $\mathcal{E}_e$ zur weiteren Verarbeitung (Zeilen 6 und 7) zwischengespeichert.

In Anlehnung an den *Hindsight Experience Replay* Ansatz [47] werden zusätzlich zu den tatsächlich gemachten Erfahrungen hypothetische Erfahrungs-Tupel $\left(s_t, a, R'_\mathrm{g}, s_{t+1}, \eta\left(\check{\sigma}_\mathcal{G}^{(g)}\right)\right)$ für alle Zielstrukturen $\mathcal{G}^{(g)}$ aus $\mathcal{G}$ dem *Replay Memory* $\mathcal{D}$ hinzugefügt. Bei der Berechnung der hypothetischen Belohnung $R'_g$ wird davon ausgegangen, dass die jeweilige Zielstruktur $\check{\sigma}_\mathcal{G}^{(g)} \in \mathcal{G}$, die während der Episode verfolgte Zielstruktur ist (Zeile 8). Dabei wird davon ausgegangen, dass diese *Augmentierung* der Trainingsdaten die Dateneffizienz des Algorithmus weiter erhöht.

## 5.3 Prozessinstantiierung

Zur Untersuchung und Evaluation der vorgestellten Methoden wurden Experimente mit einem simulierten verallgemeinerten Metallverarbeitungsprozess durchgeführt. Strukturen $\sigma$ liegen hierbei in Form der kristallographischen Textur vor. In 5.3.1 wird die verwendete Prozesssimulation erläutert. In 5.3.2 wird ausgehend davon der Aktionsraum $A$ sowie der Zustandsraum $S$ des Markov-Entscheidungsprozesses spezifiziert. Grundlage der Belohnungsfunktion ist eine Distanzfunktion $d_\sigma$ im Raum der Strukturen und eine Strukturbeschreibung $\eta(\sigma)$. Beides wird in 5.3.3 für den vorliegenden Fall kristallographischer Texturen erläutert. Anschließend werden in 5.3.4 Details zur Implementierung erläutert und in 5.3.5 die zur Evaluation verwendeten Parameter und Hyperparameter der Algorithmen spezifiziert.

### 5.3.1 Deformationsprozess

Die Prozesssimulation basiert auf dem in [8] vorgestellten *mean-field* Materialmodell. Dieses wurde erstellt nach der Beschreibung von [101] auf Basis der Taylor-Annahme [102]. Ein Prozesspfad besteht aus bis zu $K$ aufeinanderfolgenden Prozessschritten mit einachsiger Belastung in Form von Zug oder Kompression des Materials in eine beliebige Richtung. In jedem Schritt wird eine Deformation in Form des Deformationsgradienten $\tilde{\boldsymbol{F}}$

$$\tilde{\boldsymbol{F}} = \begin{bmatrix} \tilde{F}_{11} & 0 & 0 \\ 0 & F_{22} & 0 \\ 0 & 0 & F_{33} \end{bmatrix}. \tag{5.16}$$

in einer durch die Rotationsmatrix $\boldsymbol{R}$ definierten Orientierung aufgebracht. Die resultierende Deformation $\hat{\boldsymbol{F}}$ in Relation zu dem Referenzkoordinatensystem des Materialmodells ist folglich definiert als

$$\hat{\boldsymbol{F}} = \boldsymbol{R}\tilde{\boldsymbol{F}}\boldsymbol{R}^{\mathsf{T}}. \tag{5.17}$$

Zu optimierende Variablen sind $\tilde{F}_{11}$ und $\mathbf{R}$. $F_{22}$ und $F_{33}$ werden innerhalb der Simulation iterativ angepasst, bis die Spannungen im Gleichgewicht sind. Um unrealistisch hohe Deformationen zu vermeiden, wird die Prozessausführung nach der Überschreitung einer maximalen Vergleichsdehnung von 70% abgebrochen und der Prozesspfad beendet.
In Abhängigkeit der sich einstellenden kristallographischen Textur können Materialeigenschaften berechnet werden. Als exemplarische, mit geringem Aufwand zu berechnende Materialeigenschaften betrachten wir zur Untersuchung der vorgestellten Methoden die Elastizitätsmoduli $E_{ii}$ in $ii$-Richtung bezüglich des Referenzkoordinatensystems für $i \in 1,2,3$. Das Materialmodell wurde für die Untersuchungen an *DC04 Stahl* kalibriert [8].

## 5.3.2 Entscheidungsprozess

Die Prozessaktion $a_t$ in Zeitschritt $t$ des zugehörigen Entscheidungsprozesses setzt sich zusammen aus der Ausprägung der uniaxialen Deformation $f_t \in [-1,1]$ und der Orientierung der Deformation $\mathbf{q}_t \in \mathbb{R}^4, \|\mathbf{q}_t\|_2 = 1$. Die Ausprägung der Deformation $f_t$ entspricht $\tilde{F}_{11}$ im Zeitschritt $t$. Die durch das Einheitsquaternion $\mathbf{q}_t$ beschriebene Orientierung entspricht der durch $\mathbf{R}$ in 5.17 vorgegebenen Orientierung der Deformation. Da die vorgeschlagenen Lösungsmethoden einen diskreten Aktionsraum voraussetzen, ist eine Diskretisierung der Teilaktionen $f_t$ und $\mathbf{q}_t$ notwendig. Zur Untersuchung der Lösungsmethoden werden die Teilaktionen deshalb folgendermaßen definiert: $f_t \in \{0.02, -0,02\}$ und $\mathbf{q}$, als Element einer Menge $B_{100}$, bestehend aus 100 Einheitsquaternionen, die annähernd gleichmäßig verteilt den gesamten Orientierungsraum abdecken.

Darüber hinaus umfasst der Aktionsraum eine neutrale Aktion, die $\tilde{F}_{11} = 0$ entspricht und durch den Agenten genutzt werden kann, wenn keine weitere Verbesserung bezüglich der Distanz zur Zieltextur während der Episode erwartet wird. $A$ besteht folglich aus insgesamt $|A| = 2 \times |B_{100}| + 1 = 201$ Aktionen.
Zustandsbeschreibungen $\mathbf{s}$ des Entscheidungsprozesses bestehen aus Beschreibungen der durch den Prozess erreichbaren Material-Strukturen und zusätzlichen Prozessgrößen. Wie beschrieben wird die Material-Struktur $\sigma$ in dem Anwendungsfall durch die zugehörige Orientierungsdichteverteilungsfunktion (engl. *Orientation Distribution Function*, ODF, siehe Anhang A.0.1) repräsentiert. Neben der im nächsten Unterabschnitt beschriebenen numerischen Beschreibung der ODF enthält $\mathbf{s}$ zusätzliche Prozessgrößen in Form des aktuellen Zeitschritts $t$ und der aktuellen Vergleichsdehnung.

### 5.3.3 Mikrostruktur-Repräsentation und -Distanz

Zur Beschreibung $\eta(\sigma)$ der Textur $\sigma$ werden Koeffizienten der symmetrisierten *generalized Spherical Harmonics* (GSH, siehe Anhang A.0.1) $\zeta_L^\Omega$ genutzt. GSH Koeffizienten stellen eine kompakte Beschreibung von Texturen dar und werden deshalb häufig zur Beschreibung von Texturen im Kontext des maschinellen Lernens verwendet [83–85]. Die Reihe wird an dem Grad $L = 8$ abgeschnitten, sodass die Struktur des angenommenen Materials mit kubischer Kristallsymmetrie durch 21 komplexwertige GSH-Koeffizienten beschrieben wird. Die komplexwertigen Koeffizienten werden zur Verwendung als Zustandsbeschreibung in ihre Real- und Imaginärteile zerlegt, und zusammen mit den beiden zusätzlichen Prozessgrößen durch einen reellen Zustandsvektor $\mathbf{s} \in \mathbb{R}^{44}$ repräsentiert.
Die Anwendung der oben vorgestellten Methoden des bestärkenden Lernens setzen außerdem eine Distanzfunktion $d_\sigma$ im Strukturraum voraus. In der Literatur konnte keine überzeugende Methode zur Distanzberechnung auf Basis der GSH Koeffizienten gefunden werden. Zur Berechnung von Distanzen im Raum

der Texturen wurden im Rahmen dieser Arbeit deshalb eine Methode zur diskretisierten Darstellung der ODF und die Anwendung einer Histogramm-Distanzfunktion zur Ermittlung der ODF-Distanz vorgeschlagen und untersucht. Hierzu wird die ODF in $SO(3)$ diskretisiert und als normiertes Histogramm der Orientierungen dargestellt. Diese Abbildung ist nicht Topologie erhaltend und weist die Belegung der Histogramm-Klassen einem Vektor $h_\sigma \in \mathbb{R}^J$ zu, der im Folgenden als Orientierungshistogramm bezeichnet wird.
In Abhängigkeit des vorliegenden Kristallsystems $\Omega$ existieren äquivalente Orientierungen in $SO(3)$ und die ODF kann durch Auflösung dieser Äquivalenzen in einer von $\Omega$ abhängigen *Fundamentalregion* beschrieben werden. Zur Diskretisierung der ODF wird eine Menge $\mathcal{B}_J^\Omega$, bestehend aus $J$ Basisorientierungen $b_j \in \mathcal{B}_J^\Omega$ genutzt, für die angenommen wird, dass sie gleichverteilt in der *Fundamentalregion* bzgl. $\Omega$ vorliegen. Des Weiteren wird eine Distanzmetrik $\phi : SO(3) \times SO(3) \to \mathbb{R}_0^+$ vorausgesetzt. Auf dieser Basis kann eine Orientierung $\lambda \in SO(3)$ der nächsten Basisorientierung $b_j$ zugewiesen werden.
für eine einzelne Kristallorientierung $\lambda$ ist die $j$-te Komponente des binären Zuweisungsvektors $\mathbf{w}_\lambda$ definiert als

$$w_{\lambda,j} = \begin{cases} 1 & \text{, if } b_j = \arg\min_{b \in \mathcal{B}_J^\Omega}[\phi_\Omega(b,\lambda)] \\ 0 & \text{, else,} \end{cases} \tag{5.18}$$

wobei $\phi_\Omega : SO(3) \times SO(3) \to \mathbb{R}_0^+$ die minimale Distanz aller äquivalenten Orientierungen bezüglich $\Omega$ darstellt

$$\phi_\Omega(b,\lambda) = \min_{(\hat{b},\hat{\lambda}) \in \Psi_\Omega(b) \times \Psi_\Omega(\lambda)} \phi(\hat{b},\hat{\lambda}). \tag{5.19}$$

Die Funktion $\Psi_\Omega(\lambda)$ bildet dabei auf die Menge aller äquivalenten Orientierungen von $\lambda$ bezüglich $\Omega$ ab. Jeder Basisorientierung $b_j \in \mathcal{B}_J^\Omega$ ist eine *Voronoi-Zelle* in $SO(3)$ bezüglich $\phi_\Omega$ zugeordnet und der Orientierungsraum $SO(3)$ ist

durch die Gesamtheit der *Voronoi-Zellen* partitioniert. Als Orientierungshistogramm wird hier ein Vektor der relativen Häufigkeiten der Orientierungen einer ODF in den $J$ *Voronoi-Zellen* bezeichnet.
Für eine repräsentative Stichprobe $\Lambda$ von Orientierungen $\lambda$ der kristallographischen Textur $\sigma$, die zugehörigen Volumen $V(\lambda)$ und das totale Volumen der Stichprobe $V$ stellt der Vektor

$$\mathbf{h}_\sigma = \frac{1}{V} \sum_{\lambda \in \Lambda} V(\lambda) \cdot \mathbf{w}_\lambda . \tag{5.20}$$

Das Orientierungshistogramm der Textur $\sigma$ dar.
Für die kristallographischen Texturen $\sigma_a$ und $\sigma_b$ ist dann die Distanz $d_\sigma$ definiert durch das $\chi^2$-Distanzmaß, angewendet auf die zugehörigen Orientierungshistogramme $\mathbf{h}_{\sigma_a}$ und $\mathbf{h}_{\sigma_b}$:

$$\chi^2(\mathbf{h}_{\sigma_a}, \mathbf{h}_{\sigma_b}) = \sum_{j=0}^{J} \frac{(\mathbf{h}_{\sigma_a,j} - \mathbf{h}_{\sigma_b,j})^2}{(\mathbf{h}_{\sigma_a,j} + \mathbf{h}_{\sigma_b,j})}. \tag{5.21}$$

Insbesondere bei scharfen Texturen ist eine Glättung der Distanzfunktion vorteilhaft. Hierzu wird eine generalisierte Form der Zuweisung aus (5.18) genutzt. Die $i$-te Komponente des geglätteten Zuweisungsvektors $\tilde{\mathbf{w}}_\lambda$ ist definiert als

$$\tilde{w}_{\lambda,i} = \begin{cases} \dfrac{\phi_\Omega(b_i, \lambda)}{\sum_{b_j \in N} \phi_\Omega(b_j, \lambda)} & \text{, if } b_i \in N \\ 0 & \text{, else,} \end{cases} \tag{5.22}$$

wobei $N = \mathrm{NN}_k(B_J^\Omega, \lambda, \phi_\Omega)$ die Menge der $k$ nächsten Nachbarn von $\lambda$ aus der Menge $B_J^\Omega$ bezüglich des Distanzmaßes $\phi_\Omega$ ist.
Als Basisdistanz dient

$$\phi(\lambda^{(a)}, \lambda^{(b)}) = \min\Big( \Big\| q(\lambda^{(a)}) - q(\lambda^{(b)}) \Big\|_2, \Big\| q(\lambda^{(a)}) + q(\lambda^{(b)}) \Big\|_2 \Big), \tag{5.23}$$

wobei $q(\lambda) = \mathbf{q}$ auf die Darstellung der Orientierung $\lambda$ als Einheitsquaternion $\mathbf{q} \in \mathbb{R}^4, \|\mathbf{q}\|_2 = 1$ abbildet. Die Basisdistanz $\phi$ ist eine Metrik in $SO(3)$ [103]. Um eine Menge approximativ gleichverteilter Basisorientierungen $\mathcal{B}_J^\Omega$ zu erzeugen, wird der Optimierungsansatz von Quey et al. [104] genutzt. Die nächste-Nachbar Suche zur Lösung von (5.18) und (5.22) kann durch Suche in einer *k-d Baum* Struktur [105] effizient durchgeführt werden.

Zusammenfassend wird die Distanz zweier Texturen ermittelt, indem die Orientierungsdichten durch Orientierungshistogramme repräsentiert werden und die Histogramm-Distanzfunktion $\chi^2$ auf diese angewandt wird. Bei der Erstellung der normierten Histogramme sind zwei Parameter von Bedeutung, die Einfluss auf die Genauigkeit der Repräsentation sowie die resultierende Distanzfunktion haben: Die Anzahl der verwendeten Basisorientierungen $J$ und der Glättungsparameter $k$.

Die Anzahl der *Voronoi-Zellen* $J$ für die das Orientierungshistogramm gebildet wird beeinflusst den Diskretisierungsfehler der Histogramm-Darstellung. Je kleiner $J$ ist, desto größer sind die Distanzen bei der Zuweisung der *Voronoi-Zellen* in (5.18). Gleichzeitig führen hoch gewählte Kardinalitäten $J$ dazu, dass das resultierende Orientierungshistogramm zunehmend dünn besetzt ist, was zu Schwierigkeiten bei der Distanzermittlung mittels $\chi^2$ (5.21) führen kann, da diese auf dem Vergleich der Belegung der *Voronoi-Zellen* $\mathbf{h}_{\sigma,j}$ beruht. Des Weiteren hängen die rechnerischen Kosten bei der nächste-Nachbar Suche von $J$ ab[3].

[3] Bei der Verwendung von *k-d Bäumen* betragen die mittleren Kosten für die Suche nach einem nächsten Nachbar bei $J$ zufällig verteilten Punkten $O(\log J)$ [105]

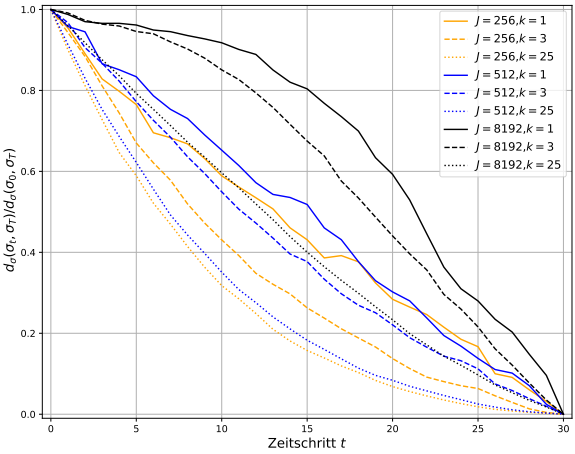



Abbildung 5.5: Experimentelle Untersuchung der eingeführten Distanz für unterschiedlich parametrisierte Orientierungshistogramme. Untersucht wird die Auswirkung verschiedener Kombinationen der Parameter $J$ und $k$ auf den Verlauf der Distanz von Strukturen $\sigma_t$ in Zeitschritt $t$ Zur Struktur im letzten Zeitschritt $\sigma_T$ für einen Pfad des eingeführten Prozesses.

Die in (5.20) eingeführte Abbildung der ODF auf ein Orientierungshistogramm $\mathbf{h}$ ist nicht Topologie erhaltend. Hierdurch kann es vorkommen, dass für ähnliche ODFs eine hohe $\chi^2$-Distanz ermittelt wird. Bei der kontinuierlichen Veränderung einer Textur, beispielsweise durch Einwirkung des oben beschriebenen Deformationsprozesses, entstehen außerdem Unstetigkeitsstellen im Verlauf der Distanz zu einer zweiten Textur. Beide Effekte lassen sich verdeutlichen, indem man den Extremfall zweier Texturen $\sigma^{(a)}$ und $\sigma^{(b)}$ betrachtet, deren Kristalle jeweils gleichgerichtet sind, wobei alle Kristalle der Textur $\sigma^{(a)}$ die Orientierung $\lambda^{(a)}$ und alle Kristalle der Textur $\sigma^{(b)}$ die Orientierung $\lambda^{(b)}$ aufweisen. Ungeachtet der Distanz der Orientierungen $\phi(\lambda^{(a)}, \lambda^{(b)})$ ist in diesem konstruierten Fall die vorgeschlagene Mikrostrukturdistanz bei Verwendung der binären Zuweisung aus (5.18) maximal ($d_\sigma(\sigma^{(a)}, \sigma^{(b)}) = 2$), solange $\lambda^{(a)}$ und $\lambda^{(b)}$ nicht derselben *Voronoi-Zelle* zugewiesen sind und schlagartig

minimal ($d_\sigma(\sigma^{(a)}, \sigma^{(b)}) = 0$) sobald beide Orientierungen in dieselbe *Voronoi-Zelle* fallen. Eine Möglichkeit diesen Problemen auf rechnerisch effiziente Art vorzubeugen, ist die in (5.22) vorgeschlagene geglättete Zuweisung. Der Grad der Glättung hängt dabei von dem Parameter $k$ ab.

Um den Effekt der Parameter $J$ und $k$ auf das Verhalten des vorgeschlagenen Distanzmaßes experimentell zu untersuchen, wurde durch den simulierten Deformationsprozess eine Reihe von $T = 30$ aufeinanderfolgenden Deformationsschritten mit gleichbleibender Ausprägung $f = 0.02$ und gleichbleibender Deformationsrichtung $\mathbf{q} = [1,0,0,0]^\intercal$, auf eine *graue Textur* aufgebracht. Der Verlauf der Distanz $d(\sigma_t, \sigma_T)$ der Textur zum Zeitschritt $t$ zur durch die Gesamtdeformation erreichten Textur $\sigma_T$ für verschiedene $(J,k)$-Kombinationen ist in Abbildung 5.5 dargestellt. Der Wertebereich der Distanzwerte innerhalb des Texturverlaufes hängt stark von den Parametern, insbesondere von $J$, ab. Da bei der Verwendung im Kontext der Belohnungsfunktion relative Distanzwerte entscheidend sind, wurden für die Visualisierung die Wertebereiche durch die Division durch die initiale Distanz $d_\sigma(\sigma_0, \sigma_T)$ vereinheitlicht. Durch die gleichbleibende Deformation bei dem exemplarischen Prozesspfad nähern sich die Texturen während der Prozessausführung konstant $\sigma_T$ und ein bezüglich $t$ monoton absteigender Verlauf der Distanzen $d(\sigma_t, \sigma_T)$ wird erwartet. Für $k = 1$ (durchgängige Linien) ist der Distanzverlauf nicht strikt monoton und ein klarer Glättungseffekt, in Abhängigkeit des Parameters $k$, kann beobachtet werden. Die Texturen bewegen sich in dem Fall nah an der initialen grauen Textur, wodurch die oben beschriebenen Effekte nicht stark ins Gewicht fallen.

Ein Diskretisierungsfehler durch die Darstellung von Texturen als Orientierungshistogramm in Abhängigkeit von $J$ und $k$ kann indirekt quantifiziert werden, indem die Abweichungen der berechneten Materialeigenschaften vor und nach der Diskretisierung der Textur herangezogen werden. In Tabelle 5.1 sind die mittleren absoluten Abweichungen der Elastizitätsmoduli $(E_{11}, E_{22}, E_{33})$ in MPa, berechnet für unterschiedliche kristallographische Texturen dargestellt. Die hierbei verwendeten 1000 Texturen wurden zufällig aus einer Menge durch den Prozess erreichbarer Texturen gezogen.

Tabelle 5.1: Mittlere absolute Abweichung der errechneten Elastizitätsmoduli $(E_{11}, E_{22}, E_{33})$ für 1000 Texturen und zugehörigen Orientierungshistogrammen in Abhängigkeit der Parameter $J, k$ (in GPa)

| | | $k$ | | |
|---|---|---|---|---|
| | | 1 | 3 | 25 |
| $J$ | 256 | 0.574 | 0.238 | 0.401 |
| | 512 | 0.493 | 0.201 | 0.275 |
| | 8192 | 0.196 | 0.077 | 0.057 |

Bei den Ergebnissen ist ein klarer positiver Effekt der Erhöhung der Anzahl der gewählten Basisorientierungen $J$ zu sehen. Auch die Glättung wirkt sich bis zu einem gewissen Grad $k$ positiv auf die Ergebnisse aus. Die Wahl der Parameter $J$ und $k$ folgt einer Abwägung der hier vorgestellten Resultate, den oben angestellten theoretischen Überlegungen und dem negativen Einfluss von $J$ bezüglich des Laufzeitverhaltens bei der Distanzberechnung. Die Parameter die für die im Folgenden vorgestellten Experimente gewählt wurden sind $(J = 512, k = 3)$.

### 5.3.4 Implementierung

Die in 4.3.5 beschriebene Experimentalumgebung wurde für die im Folgenden vorgestellten Untersuchungen ausgeprägt. Daneben wurden insbesondere die folgenden Softwarepakete genutzt: Die Implementierungen der Basisalgorithmen des bestärkenden Lernens (*Deep Q Networks* sowie die Erweiterungen *Prioritized Experience Replay*, *Double Q-Learning*, *Dueling Q-Learning*) stammen aus der Algorithmen-Sammlung *stable baselines* [106]. Bei der Berechnung der Orientierungshistogramme wird die *SciPy* [107] Implementierung des *k-d tree* Algorithmus verwendet. Das Softwarepaket *Neper* [108] wird

verwendet, um die annähernd gleichmäßig verteilten Orientierungen nach dem Optimierungsansatz aus [104] zu generieren. Polfiguren in den Abbildungen aus Abschnitt 5.4 wurden mithilfe des MTEX Frameworks erstellt [109].

### 5.3.5 Versuchsaufbau, Netzarchitekturen und Parameter

Zur Untersuchung und Evaluation der vorgestellten Methoden werden in aufeinanderfolgenden Episoden Pfade des in Abschnitt 5.3 vorgestellten Prozesses ausgeführt. Als maximale Pfadlänge pro Episode wurde dabei $T = 100$ festgelegt. Jeder Prozesspfad startet mit einer annähernd gleichmäßig verteilten ODF (im Folgenden als *graue Textur* bezeichnet), repräsentiert durch 250 gleichgewichteten Kristallorientierungen, die annähernd gleichmäßig verteilt in der *Fundamentalregion* bezüglich der kubischen Kristallsymmetrie des DC04 Stahls liegen.
Die Experimente zur Untersuchung der Struktur-geleiteten Optimierung mit **einzelnen Zieltexturen** bestehen aus Optimierungsläufen mit jeweils 98 aufeinanderfolgenden Prozessepisoden. Sofern bei der Vorstellung der Ergebnisse nicht anders angegeben, wurden sie mit den folgenden Parametern SG-SGPPO Algorithmus, Hyperparametern der künstlichen Neuronalen Netze und Einstellungen durchgeführt:

- Die Orientierungshistogramme zur Berechnung der $\chi^2$-Distanz werden mit den Parametern $J = 512$, $k = 3$ erzeugt.
- Der Entscheidungsprozess ist nicht diskontiert, $\gamma = 1$.
- Als Basisalgorithmus dient DQN mit den Erweiterungen *Prioritized Experience Replay*, *Double Q-Learning*, *Dueling Q-Learning*.
- Das Update der Parameter $\theta^-$ findet alle $n_\theta = 250$ Zeitschritte statt.

- *Prioritized Experience Replay* wird mit den Parametern $\alpha_{PER} = 0.6, \beta_0 = 0.4$ angewandt.
- Die Lernstrategie $\tilde{\pi}$ ist $\varepsilon$-*greedy*, die initiale Explorationsrate beträgt $\varepsilon_0 = 0.5$, die finale Explorationsrate $\varepsilon_\mathrm{f} = 0.1$, $n_\varepsilon = 50$.
- Die verwendeten künstlichen Neuronalen Netze sind *feedforward Netze* mit drei versteckten Schichten, ReLU Aktivierungsfunktionen und *layer normalization* [110]. Von der auf die Eingabeschicht ausgehend beträgt die Breite der versteckten Schichten $(128, 64, 32)$.
- Das Training der Netze beginnt nach den ersten 100 Zeitschritten. Netze werden mit *mini-Batches* bestehend aus 32 Samples trainiert.
- Als Optimierungsalgorithmus wird ADAM mit einer Lernrate von $5e^{-4}$ verwendet.

Die Experimente zur Untersuchung der Struktur-geleiteten Optimierung mit **einer Menge äquivalenter Zieltexturen** unterscheidet sich in folgenden Punkten:

- Ein Optimierungslauf besteht jeweils aus $n_e = 200$ aufeinanderfolgenden Episoden.
- Die *feedforward Netze* bestehen aus 4 versteckten Schichten der Breite $(128, 256, 256, 128)$, aufgrund der höheren Komplexität der Abbildung durch die Generalisierung über Zieltexturen.
- Pro Zeitschritt wird mit vier *mini-Batches* trainiert.
- Für eine verbesserte Stabilität des Lernvorgangs findet das Update der Parameter $\theta^-$ alle $n_\theta = 500$ Zeitschritte statt.
- Parameter der $\varepsilon$-*greedy* Lernstrategie $\tilde{\pi}$ sind $\varepsilon_0 = 0.5, \varepsilon_\mathrm{f} = 0, n_\varepsilon = 190$. Die $\varepsilon_\mathrm{G}$-*greedy* Auswahl der Zielstruktur pro Episode findet mit folgenden Parametern statt: $\check{\varepsilon}_0 = 1$, $\check{\varepsilon}_f = 0$, $n_{\check{\varepsilon}} = 190$.

Die Experimente zur Untersuchung der Ansätze wurden auf Workstations mit jeweils 20 2.2 GHz CPU-Kernen und einer GTX 1080 Ti GPU durchgeführt. Während Berechnungen des Simulationsframeworks auf der CPU stattfinden, findet die GPU Anwendung bei dem Training der künstlichen Neuronalen Netze.

## 5.4 Ergebnisse

Die hier vorgestellten Evaluationsergebnisse wurden experimentell, auf Basis der in Abschnitt 5.3 dargestellten Prozess-Instanziierung, erzeugt. Dieser Abschnitt ist zwei-geteilt. In dem Unterabschnitt 5.4.1 werden Evaluationsergebnisse für den in 5.2.1 vorgestellten SG-SGPPO Algorithmus zur Optimierung mit einzelnen Zielstrukturen vorgestellt. Unterabschnitt 5.4.2 behandelt Ergebnisse der Evaluation des in 5.2.2 vorgestellten MEG-SGPPO Algorithmus zur Optimierung mit mehreren äquivalenten Zielstrukturen.

### 5.4.1 Einzelne Zielmikrostrukturen

Der in 5.2.1 vorgestellte SG-SGPPO Ansatz zur Optimierung des Prozesspfades zur Erreichung einzelner Material-Strukturen wurde für sechs unterschiedliche zufällig ausgewählte Zieltexturen $\Gamma = \{\breve{\sigma}^{(0)}, \breve{\sigma}^{(1)}, \breve{\sigma}^{(2)}, \breve{\sigma}^{(3)}, \breve{\sigma}^{(4)}, \breve{\sigma}^{(5)}\}$ evaluiert. Diese erfüllen folgende Kriterien:

1. $d_\sigma(\breve{\sigma}^{(i)}, \breve{\sigma}^{(j)}) > 1.2$ für alle $\breve{\sigma}^{(i)}, \breve{\sigma}^{(j)} \in \Gamma$ zur Sicherstellung der Diversität.

2. einem Mindestabstand $\min_{\breve{\sigma}\in\Gamma}\left(d_\sigma(\breve{\sigma}, \sigma_0)\right) > 0.75$, wobei $\sigma_0$ die Ausgangstextur, in diesem Fall die *graue Textur*, darstellt.

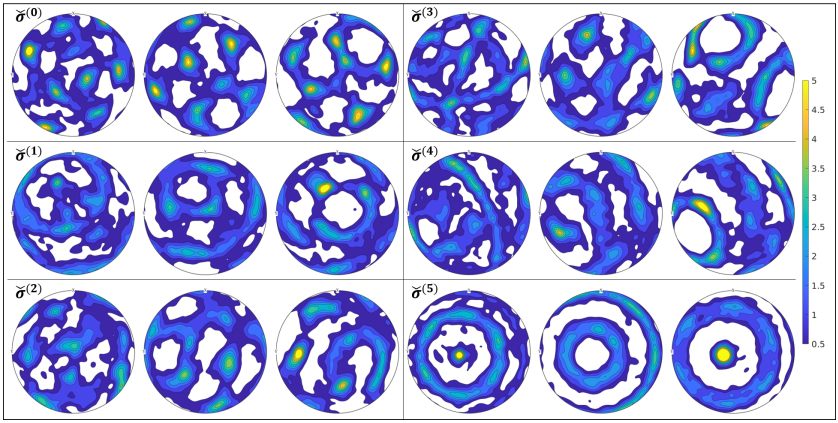



Abbildung 5.6: Zieltexturen aus Γ, jeweils dargestellt durch die $(001)$-,$(011)$-,$(111)$-Polfiguren.

Die Polfiguren der sechs Zieltexturen sind in Abbildung 5.6 dargestellt. Texturen werden in dieser Arbeit als Polfiguren der Miller-Indizes $(100), (110), (111)$ dargestellt.
Die zugehörigen, der Beschreibung in 5.3.1 folgend berechneten Elastizitätsmoduli sind in Tabelle 5.2 angeführt.

Tabelle 5.2: Elastizitätsmoduli der Texturen aus Γ (in GPa)

| Goal | $E_{11}$ | $E_{22}$ | $E_{33}$ |
|---|---|---|---|
| $\breve{\sigma}^{(0)}$ | 221 | 223 | 221 |
| $\breve{\sigma}^{(1)}$ | 216 | 221 | 212 |
| $\breve{\sigma}^{(2)}$ | 223 | 219 | 214 |
| $\breve{\sigma}^{(3)}$ | 222 | 218 | 223 |
| $\breve{\sigma}^{(4)}$ | 224 | 218 | 219 |
| $\breve{\sigma}^{(5)}$ | 227 | 226 | 233 |

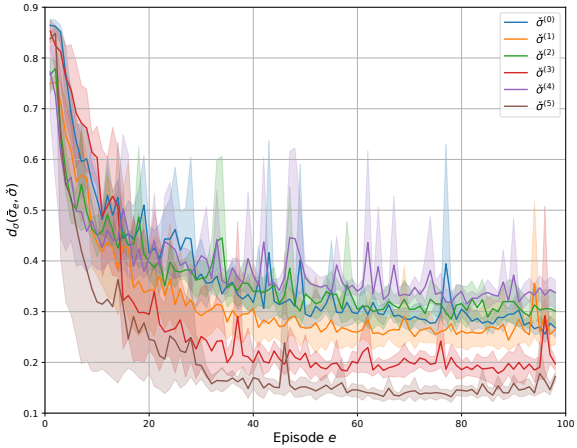



Abbildung 5.7: Distanz der jeweils nächsten erreichten Textur $\tilde{\sigma}_e$ zur Zieltextur $\check{\sigma}^{(g)} \in \Gamma$ pro Episode $e$. Dargestellt ist jeweils der Mittelwert und das 95% Konfidenzintervall für 5 unabhängige Optimierungsläufe.

Der Optimierungs-Algorithmus wurde für jede Textur aus $\Gamma$ separat angewandt. Aufgrund der stochastischen Initialisierung der Netzparameter und der stochastischen Wahl der Aktionen bei Anwendung der Lernstrategie $\tilde{\pi}$ variieren die Ergebnisse des Algorithmus. Deshalb wurden jeweils mehrere unabhängige Optimierungsläufe durchgeführt.

Ein Prozesspfad besteht aus maximal $T$ Prozessschritten. Eine Episode des bestärkenden Lernens endet nach $T$ Prozessschritten oder bereits vorher, falls die maximale Vergleichsdehnung erreicht ist. Bei der Evaluation wird davon ausgegangen, dass der optimale Prozesspfad im Vorfeld der finalen Prozessausführung ermittelt wird. Dies impliziert, dass in einer Episode $e$ auch jeder Teilpfad, der die initiale Textur $\sigma_0$ mit einer Textur $\sigma_t \in \Sigma_e$ verbindet, ein gültiges Optimierungsergebnis darstellt. Die Menge $\Sigma_e \subset \Sigma$ umfasst die während Episode $e$ erreichten Texturen. Der beste in Episode $e$ ermittelte Teilpfad verbindet $\sigma_0$ mit

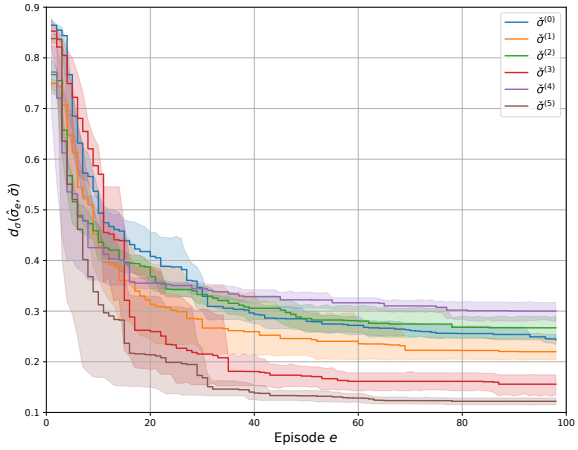



Abbildung 5.8: Distanz der in den bisherigen Episoden besten erreichten Struktur $\hat{\sigma}$ zur Zieltextur $\check{\sigma}^{(g)} \in \Gamma$ pro Episode $e$. Dargestellt ist jeweils der Mittelwert und das 95% Konfidenzintervall für 5 unabhängige Optimierungsläufe.

$$\tilde{\sigma}_e = \underset{\sigma \in \Sigma_e}{\arg\min} \left( d_\sigma(\sigma, \check{\sigma}) \right). \tag{5.24}$$

Ziel der Optimierung ist dann die Minimierung der Distanz $d_\sigma(\tilde{\sigma}_e, \check{\sigma})$. Pro Zieltextur in $\Gamma$ wurden fünf unabhängige Optimierungsläufe, jeweils aus 98 Episoden bestehend, durchgeführt. In Abbildung 5.7 ist die mittlere Distanz $d_\sigma(\tilde{\sigma}_e, \check{\sigma})$ in Abhängigkeit der Episode $e$ und das zugehörige 95% Konfidenzintervall (ermittelt aus den Werten der fünf Optimierungsläufe) pro Zieltextur dargestellt.

Bei der Optimierung im Vorfeld der finalen Prozessausführung ist insbesondere die Qualität des Optimierungsergebnisses in Abhängigkeit von dem Optimierungsaufwand ausschlaggebend. Es genügt daher die Betrachtung des bis zu einer Episode gefundenen besten Ergebnisses aller bisherigen Episoden. Deshalb wird im Folgenden ein weiterer Aggregationsschritt angewandt, indem für Episode $e$ die Distanz des bisher besten Ergebnisses betrachtet wird. Der nach

Episode $e$ beste in den bisherigen Episoden gefundene Prozesspfad verbindet $\sigma_0$ mit

$$\hat{\sigma}_e = \underset{i\in[0,e]}{\arg\min} \left[ d_\sigma \left[ \underset{\sigma\in\Sigma_i}{\arg\min} \left( d_\sigma(\sigma, \check{\sigma}) \right), \check{\sigma} \right] \right]. \tag{5.25}$$

Die zugehörige Distanz $d_\sigma(\hat{\sigma}_e, \check{\sigma})$ ist in Abbildung 5.8 in Abhängigkeit der Episode $e$ als Mittelwert und 95% Konfidenzintervall für die unabhängigen Optimierungsläufe pro Zieltextur $\check{\sigma} \in \Gamma$ dargestellt.
Die ermittelten Distanzen der besten Optimierungsergebnisse $d_\sigma(\hat{\sigma}_{98}, \check{\sigma})$ pro $\check{\sigma} \in \Gamma$ für die unabhängigen Optimierungsläufe liegen in den Intervallen:

- $[0.2330, 0.2591]$ für $\check{\sigma} = \check{\sigma}^{(0)}$,
- $[0.2010, 0.2689]$ für $\check{\sigma} = \check{\sigma}^{(1)}$,
- $[0.2354, 0.3040]$ für $\check{\sigma} = \check{\sigma}^{(2)}$,
- $[0.1165, 0.1809]$ für $\check{\sigma} = \check{\sigma}^{(3)}$,
- $[0.2656, 0.3283]$ für $\check{\sigma} = \check{\sigma}^{(4)}$,
- $[0.1109, 0.1327]$ für $\check{\sigma} = \check{\sigma}^{(5)}$.

Qualitative Ergebnisse in Form von Polfiguren der Ergebnis-Texturen zu verschiedenen Zeitpunkten der Optimierung sind in 5.9 dargestellt. Die dargestellten Ergebnisse stammen aus einem der fünf unabhängigen Optimierungsläufe mit der Zieltextur $\check{\sigma}^{(0)}$. Abbildung 5.9 stellt für den untersuchten Optimierungslauf die Distanz $d_\sigma(\hat{\sigma}_e, \check{\sigma}^{(0)})$ im Verlauf der Episoden $e$ dar. Zusätzlich sind Texturen $\hat{\sigma}_e$ für einzelne Episoden $e \in \{0, 4, 10, 21, 40, 97\}$, sowie die Zieltextur $\check{\sigma}^{(0)}$ in Form von Polfiguren dargestellt. Anhand dieser qualitativen Darstellungen kann die Annäherung der Ergebnisse $\hat{\sigma}_e$ an die Zieltextur $\check{\sigma}^{(0)}$ über den Optimierungsverlauf hinweg nachvollzogen werden. Das beste Ergebnis in dem Optimierungslauf wird in Episode 97 erreicht ($\hat{\sigma}_{98} = \tilde{\sigma}_{97}$). In Abbildung 5.10 ist der Prozesspfad aus Episode 97 visualisiert. Die dargestellten Punkte geben dabei pro Zeitschritt $t$ die gewählte Prozessaktion an, wobei die Farbe

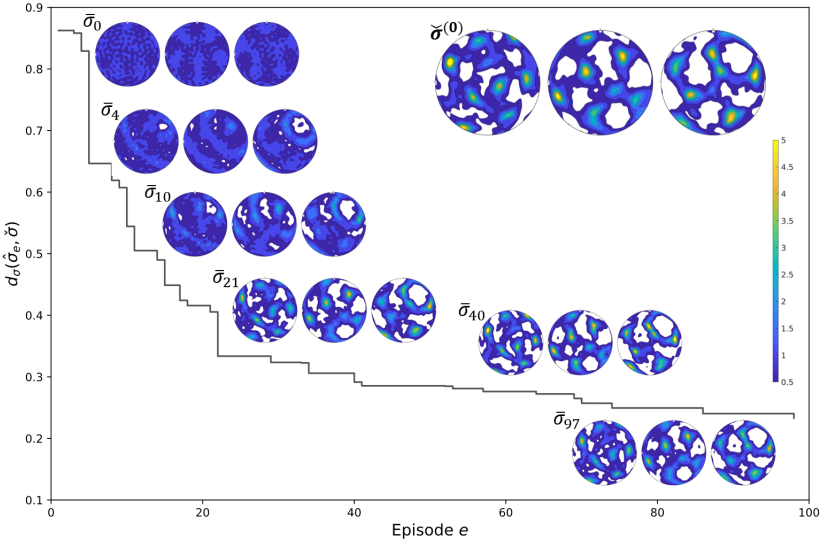



Abbildung 5.9: Detailansicht eines einzelnen Optimierungslaufs mit der Zielstruktur $\check{\sigma}^{(0)}$. Kombinierte Darstellung bestehend aus dem Verlauf der Distanz $d_\sigma(\hat{\sigma}_e, \check{\sigma}^{(0)})$ der bisher besten erreichten Struktur $d_\sigma(\hat{\sigma}_e$ zur Zielstruktur $\check{\sigma}^{(0)}$ und der qualitativen Darstellung einzelner erreichter Strukturen $\bar{\sigma}_e$ und der Zielstruktur $\check{\sigma}^{(0)}$ in Form von (001)-,(011)-,(111)-Polfiguren.

der Punkte die Art der Deformation $f_t$ und die vertikale Lage die Orientierung der Deformation $\mathbf{q}_t \in o_{100}$ (siehe 5.3.2) entsprechend der linken Koordinatenachse darstellt. Die Distanz $d_\sigma(\sigma_t, \check{\sigma}^{(0)})$ in Abhängigkeit von $t$ für Episode 97 ist als Linienplot durch den Ordinatenwert entsprechend der rechten Koordinaten-Achse dargestellt. Der optimale Teilpfad endet mit der kleinsten Distanz zur Zielstruktur innerhalb der Episode zum Zeitschritt $t = 46$ und ist in Form einer gestrichelten vertikalen Linie dargestellt. Nach Zeitschritt $t = 66$ endet Episode 97 da das Abbruch-Kriterium erreicht wird, indem die Vergleichsdehnung den Maximalwert von 70% überschreitet.
In Abbildung 5.11 sind weitere qualitative Ergebnisse dargestellt. In der oberen Reihe sind die jeweils besten gefundenen Prozesspfade in derselben Form wie

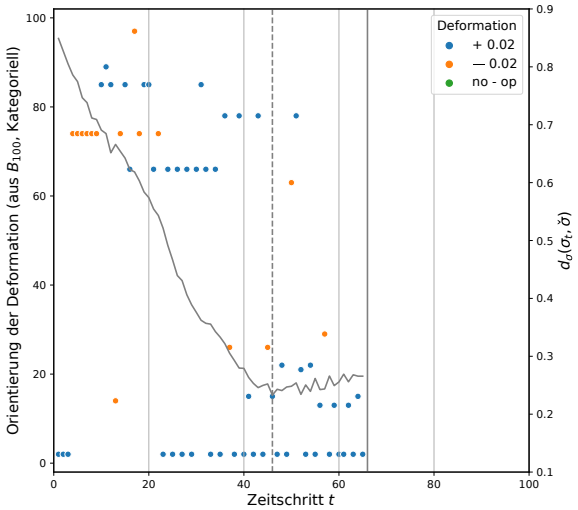



Abbildung 5.10: Detailansicht der besten Episode $e = 97$ des in Abbildung 5.9 dargestellten Optimierungslaufs. Prozessaktionen $a_t$ sind repräsentiert durch Punkte, deren Position auf der $y$-Achse die Orientierung der Deformation (linke vertikale Achsenbeschriftung) und deren Farbe die Art der Deformation $f_t$ angibt. Das Ende der Episode (hier durch erreichen der maximalen Vergleichsdehnung) ist durch eine solide vertikale Linie gekennzeichnet. Der Verlauf der Distanz $d_\sigma(\sigma_t, \breve{\sigma})$ (rechte vertikale Achsenbeschriftung) ist als Linienplot dargestellt. Der Zeitschritt der Erreichung von $\tilde{\sigma}$ ist durch eine gestrichelte graue vertikale Linie gekennzeichnet.

in Abbildung 5.10 für alle Zieltexturen aus $\Gamma$ dargestellt. Darunter befinden sich pro Zieltextur drei Zeilen mit Polfiguren:

- Obere Zeile: Polfiguren der Zieltextur $\breve{\sigma}^{(i)}$ für $i \in \{0,1,2,3,4,5\}$.
- Mittlere Zeile: Polfiguren der Textur die durch den über alle Optimierungsläufe hinweg besten ermittelten Prozesspfad erreicht wird pro Zieltextur $\breve{\sigma}^{(i)}$.
- Untere Zeile: Polfiguren der Textur die durch den im schlechtesten Optimierungslauf ermittelten Prozesspfad erreicht wird pro Zieltextur $\breve{\sigma}^{(i)}$.

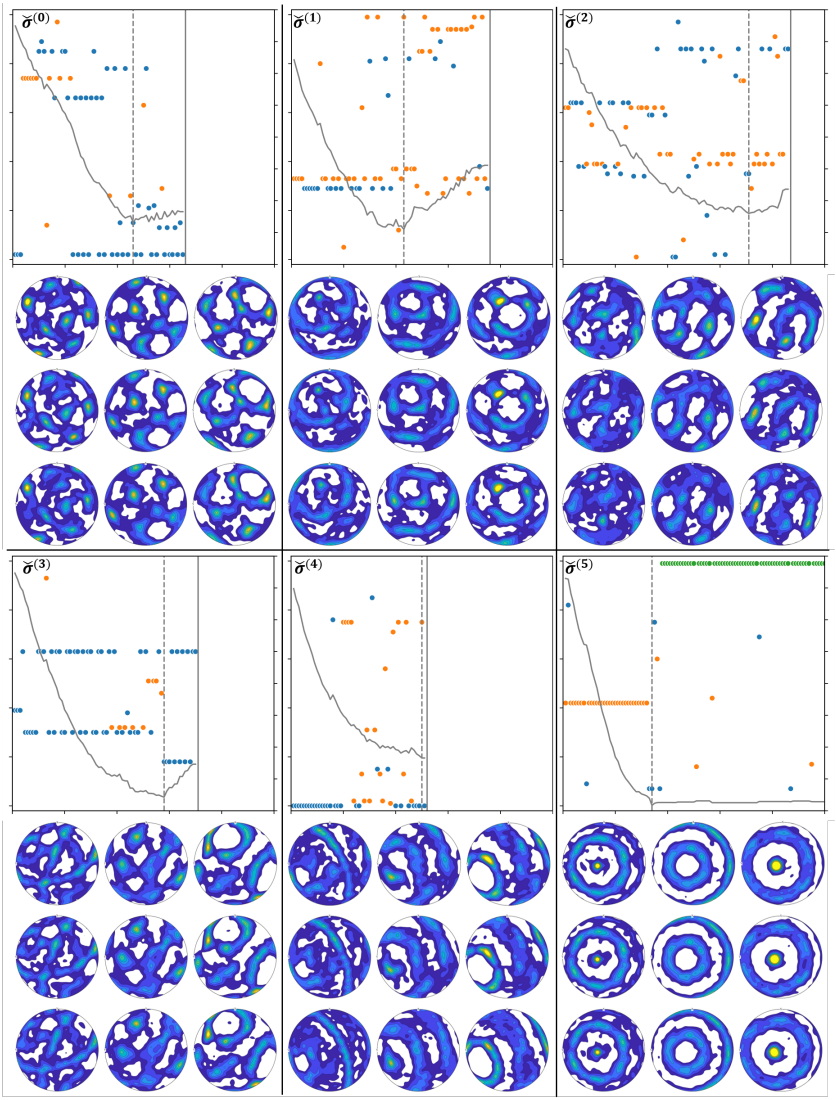



Abbildung 5.11: Ergebnisse pro Zielstruktur aus Γ. Jeweils Zeilenweise: (1) Detailansicht der besten Episode eines Optimierungslaufes. Achsenparameter und Labels entsprechen Abbildung 5.10. (2) Polfiguren der Zielstruktur. (3) Polfiguren der besten erreichten Struktur $\hat{\sigma}_{98}$ des besten Optimierungslaufs. (4) Polfiguren der im schlechtesten Optimierungslauf erreichten Struktur $\hat{\sigma}_{98}$.

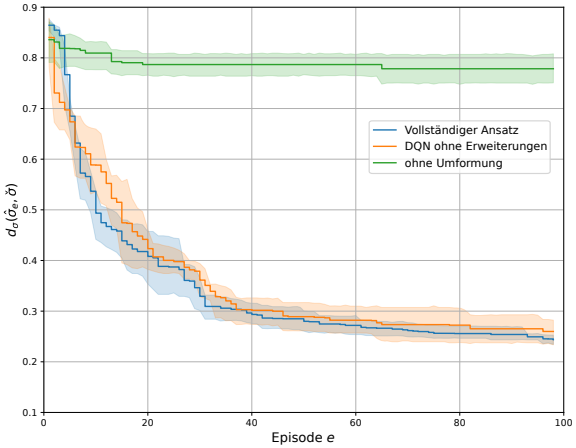



Abbildung 5.12: Ergebnisse der Ablationsstudie für einzelne Zielstrukturen. Pro Variante des Algorithmus ist der Verlauf der Distanz $d_\sigma(\hat{\sigma}, \check{\sigma}^{(0)})$ in Form des Mittelwertes und des 95% Konfidenzintervalls für fünf unabhängige Optimierungsläufe dargestellt.

Um den Einfluss einzelner methodischer Komponenten zu untersuchen wurde eine Ablationsstudie durchgeführt. Dabei wird die Performanz der Gesamtmethode mit der Performanz verglichen, die ohne das vorhandensein einzelner methodischer Komponenten erzielt wird. Relevante methodische Komponenten für den Fall mit nur einer Zieltextur sind zum einen die DQN Erweiterungen (*Prioritized Experience Replay*, *Double Q-Learning*, *Dueling Q-Learning*) und zum anderen die Umformung der Belohnungsfunktion. Ergebnisse der Ablationsstudie sind in Abbildung 5.12 dargestellt. Für jede Ausprägung basieren die dargestellten Ergebnisse auf fünf unabhängigen Optimierungsläufen mit der Zieltextur $\check{\sigma}^{(0)}$. Ergebnisse sind in Form des Mittelwerts und des 95% Konfidenzintervalls der bereits oben verwendeten besten bisherigen Zieldistanz dargestellt.

### 5.4.2 Mehrere äquivalente Zielmikrostrukturen

Bei der im vorangegangenen Unterabschnitt beschriebenen Anwendung des SG-SGPPO Algorithmus zur Pfadoptimierung im Fall einer einzelnen Zieltextur wurde keine zufriedenstellende Lösung für einige der Zieltexturen aus $\Gamma$ gefunden. Insbesondere betrifft dies die Zieltexturen $\breve{\sigma}^{(2)}$ und $\breve{\sigma}^{(4)}$ (vgl. Abbildungen 5.7 und 5.8). Wie beschrieben ist das eigentliche Ziel bei der Optimierung der Prozesspfade die Erreichung einer Material-Struktur die gewünschte Eigenschaften aufweist und, wie in Kapitel 5.1 beschrieben, existieren häufig mehrere Material-Strukturen, die bezüglich der gewünschten Materialeigenschaften äquivalent sind. Der in 5.2.2 vorgestellte MEG-SGPPO Algorithmus zur Struktur-geleiteten Optimierung mit mehreren äquivalenten Zielstrukturen identifiziert eine gut erreichbare Struktur aus der Menge der äquivalenten Strukturen und löst zeitgleich das Prozesspfadoptimierungsproblem.
Um die Vorteile der erweiterten Methode zu untersuchen, werden zwei Mengen unterschiedlicher Zieltexturen, $\mathcal{G}_{\text{4-equiv}}$ und $\mathcal{G}_{\text{2-equiv}}$, verwendet. Die Menge $\mathcal{G}_{\text{4-equiv}}$ besteht aus 10 Texturen $\breve{\sigma}^{(g)}_{\text{4-equiv}}$, die als äquivalent zu der schlecht erreichbaren Textur $\breve{\sigma}^{(4)}$ betrachtet werden. Die Menge $\mathcal{G}_{\text{2-equiv}}$ besteht aus 10 Texturen $\breve{\sigma}^{(g)}_{\text{2-equiv}}$ die als äquivalent zu der zweiten schlecht erreichbaren Textur $\breve{\sigma}^{(2)}$ betrachtet werden. Eine Textur $\breve{\sigma}^{(g)}_{i\text{-equiv}}$ wird als Äquivalent zu $\breve{\sigma}^{(i)}$ betrachtet, wenn die Elastizitätsmoduli $(E_{11}, E_{22}, E_{33})$ von $\breve{\sigma}^{(i)}$ und $\breve{\sigma}^{(g)}_{i\text{-equiv}}$ um maximal 0.5 GPa abweichen.
Der in 5.2.2 vorgestellte Algorithmus wurde für jeweils $n_e = 200$ Episoden auf die Mengen $\mathcal{G}_{\text{4-equiv}}$ und $\mathcal{G}_{\text{2-equiv}}$ angewandt. Die Optimierungsverläufe sind in Abbildung 5.13 für die Optimierung mit den Zieltexturen $\mathcal{G}_{\text{4-equiv}}$ und in Abbildung 5.14 für die Optimierung mit den Zieltexturen $\mathcal{G}_{\text{2-equiv}}$ dargestellt. Die Punkte zeigen dabei je Episode $e$ die Distanz $d_\sigma(\tilde{\sigma}_e, \breve{\sigma})$ der aus dem besten Teilpfad resultierenden Textur $\tilde{\sigma}_e$ zur Zieltextur $\breve{\sigma}$ an. Die Zieltextur $\breve{\sigma}$ wird, wie in 5.2.2 beschrieben, durch den Agenten zu Beginn jeder Episode aus $\mathcal{G}_{\text{4-equiv}}$, bzw. aus $\mathcal{G}_{\text{2-equiv}}$, gewählt. Die Farbe des Punktes repräsentiert jeweils die in Episode $e$ als Zieltextur gewählte Textur $\breve{\sigma}^{(g)}_{i\text{-equiv}} \in \mathcal{G}_{\text{i-equiv}}$. Die

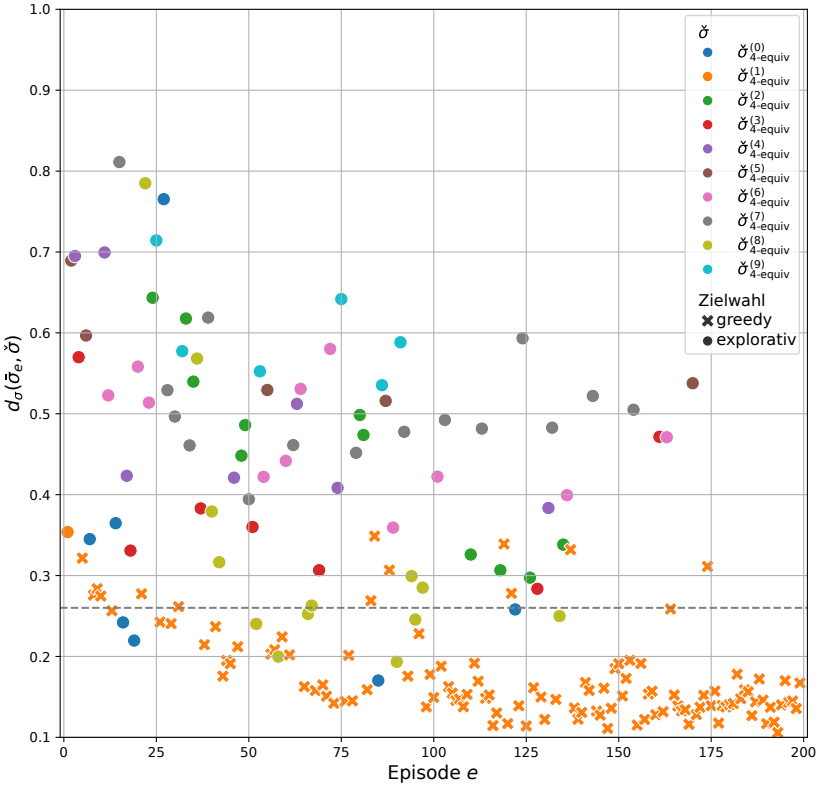



Abbildung 5.13: Verlauf der Optimierung mit mehreren äquivalenten Zielstrukturen $\mathcal{G}_{\text{4-equiv}}$. Dargestellt ist pro Episode $e$ die ausgewählte Zielstruktur $\check{\sigma}$ (Farbe der Punkte) und die Art der Wahl (Form der Punkte), sowie die Distanz der besten erreichten Struktur $\tilde{\sigma}_e$ zur gewählten Zielstruktur (vertikale Lage der Punkte).

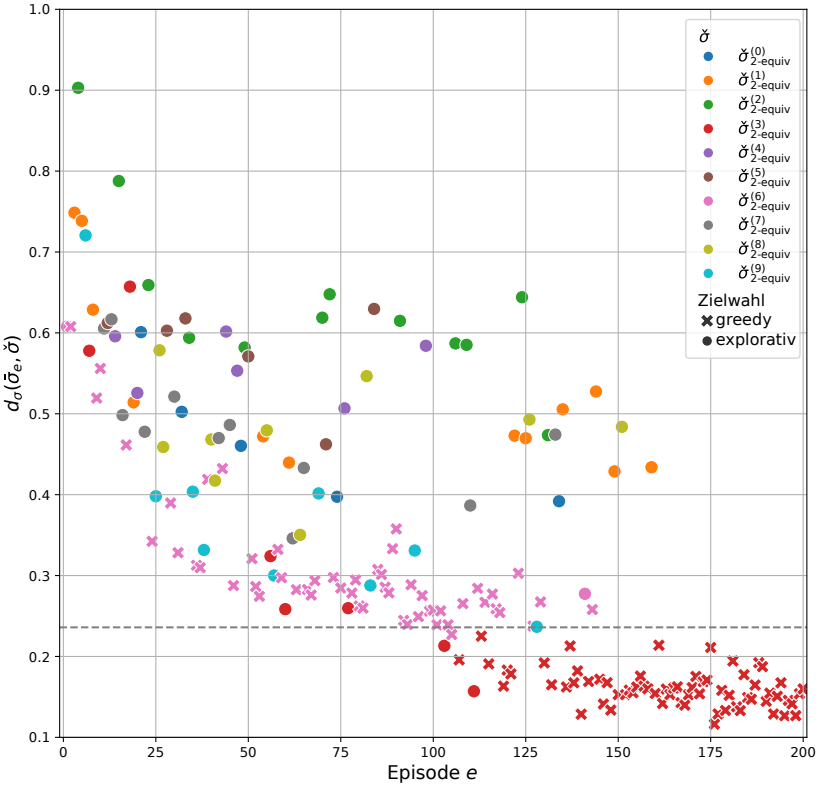



Abbildung 5.14: Verlauf der Optimierung mit mehreren äquivalenten Zielstrukturen $\mathcal{G}_{\text{2-equiv}}$. Dargestellt ist pro Episode $e$ die ausgewählte Zielstruktur $\check{\sigma}$ (Farbe der Punkte) und die Art der Wahl (Form der Punkte), sowie die Distanz der besten erreichten Struktur $\tilde{\sigma}_e$ zur gewählten Zielstruktur (vertikale Lage der Punkte).

Form der Punkte repräsentiert jeweils die Art der Entscheidung, wobei eine kreuzförmige Markierung für eine *greedy* gewählte Zieltextur (*Exploitation*) und eine runde Markierung für eine zufällig gewählte Zieltextur (Exploration) steht. Eine horizontale, gestrichelte Linie gibt jeweils die Distanz der besten Textur an, zu der ein Pfad in den vorangehend beschriebenen Versuchen für die jeweilige Zieltextur ermittelt wurde ($d_\sigma = 0.2656$ für $\check{\sigma}^{(4)}$, $d_\sigma = 0.2354$ für $\check{\sigma}^{(2)}$).

Der Explorationsfaktor der Zielstrukturwahl $\check{\varepsilon}$ beträgt zu Beginn der Optimierung $\check{\varepsilon}_0 = 1$ und nimmt während des Optimierungsverlaufs linear ab, bis er $\check{\varepsilon}_f = 0$ erreicht (siehe 5.3.5). Dies spiegelt sich in den Abbildungen wider. Während zu Beginn der Optimierung Zieltexturen überwiegend zufällig gewählt werden, legt der Agent sich im weiteren Verlauf auf eine Textur fest und optimiert den Prozesspfad zur Erreichung dieser Textur.

In der Menge der zu $\check{\sigma}^{(4)}$ äquivalenten Texturen $\mathcal{G}_{\text{4-equiv}}$ befindet sich eine Textur, $\check{\sigma}^{(1)}_{\text{4-equiv}}$, sehr nah an der grauen Ausgangstextur. Diese wird, wie in den Episoden mit *greedy*-Zieltexturwahl ersichtlich, durch den Agenten durchgängig und von Beginn an präferiert. Gleichzeitig werden in anderen Episoden auch zu anderen Zieltexturen aus $\mathcal{G}_{\text{4-equiv}}$, namentlich $(\check{\sigma}^{(0)}_{\text{4-equiv}}, \check{\sigma}^{(8)}_{\text{4-equiv}})$, Prozesspfade gefunden, die dem besten gefundenen Prozesspfad zu $\check{\sigma}^{(4)}$ überlegen sind.

Im Fall der $\check{\sigma}^{(2)}$-äquivalenten Texturen $\mathcal{G}_{\text{2-equiv}}$ verfolgt der Agent während der anfänglichen 105 Episoden das Ziel einen guten Prozesspfad für $\check{\sigma}^{(6)}_{\text{2-equiv}}$ zu finden. Nachdem in Episode 103 ein überlegener Prozesspfad für $\check{\sigma}^{(3)}_{\text{2-equiv}}$ gefunden wurde, fluktuiert die *greedy*-Wahl der Zieltextur zwischen $\check{\sigma}^{(3)}_{\text{2-equiv}}$ und $\check{\sigma}^{(6)}_{\text{2-equiv}}$. Nach Episode 144 setzt sich $\check{\sigma}^{(3)}_{\text{2-equiv}}$ aufgrund der nachhaltig besseren Ergebnisse als verfolgte Zieltextur durch.

In beiden Fällen werden durch den Algorithmus für mehrere äquivalente Zieltexturen deutlich bessere Prozesspfade als bei der Optimierung für eine einzelne Zieltextur gefunden. Im Fall von $\mathcal{G}_{\text{4-equiv}}$ werden bereits während der ersten 25 Episoden bessere Ergebnisse geliefert, im Fall $\mathcal{G}_{\text{2-equiv}}$ während der ersten 75 Episoden.

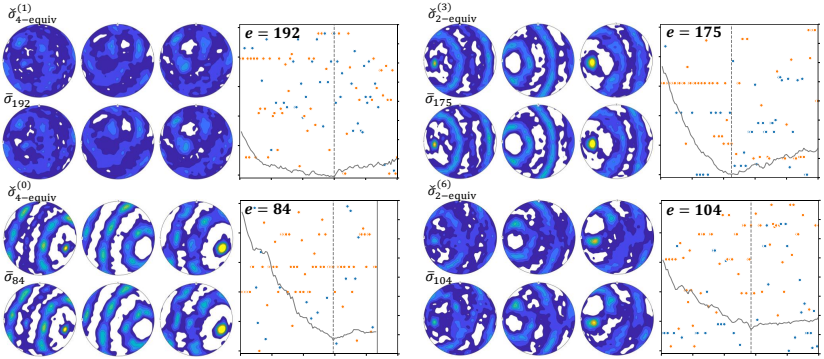



Abbildung 5.15: Qualitative Ergebnisse des in Abbildung 5.13 dargestellten Optimierungslaufs für $\mathcal{G}_{4\text{-equiv}}$ (linke Seite) und des in Abbildung 5.14 dargestellten Optimierungslaufs für $\mathcal{G}_{2\text{-equiv}}$ (rechte Seite). Jeweils: $(0,0,1),(0,1,1),(1,1,1)$ Polfiguren der zwei Zielstrukturen $\breve{\sigma}^{(g)}_{\text{i-equiv}}$ die während der Optimierung am besten angenähert wurden und der zugehörigen besten Annäherungen $\tilde{\sigma}_e$. Daneben jeweils: Eine Detaildarstellung der Episode $e$. Achsenparameter und Labels entsprechen dabei Abbildung 5.10.

Für die jeweils zwei am besten erreichten äquivalenten Texturen sind in Abbildung 5.15 qualitative Ergebnisse in Form von Polfiguren der Zieltextur und der nächsten erreichten Textur, sowie Detaildarstellungen der zugehörigen Episode dargestellt. Die am besten erreichten äquivalenten Texturen sind im Einzelnen $\breve{\sigma}^{(1)}_{4\text{-equiv}}$ und $\breve{\sigma}^{(0)}_{4\text{-equiv}}$, sowie $\breve{\sigma}^{(3)}_{2\text{-equiv}}$ und $\breve{\sigma}^{(6)}_{2\text{-equiv}}$. Die Episoden-Detaildarstellungen entsprechen im Aufbau und bezüglich der Achsen-Beschriftungen und -Skalierungen der Detaildarstellung aus Abbildung 5.10. Die Polfiguren geben jeweils einen Eindruck davon, wie gut die Zieltexturen erreicht werden konnten. Die Prozessdiagramme spiegeln den Aufbau und die Komplexität der ermittelten Prozesspfade wieder.

Der Einfluss der in 5.2.2 vorgestellten *Augmentierung* der Trainingsdaten wurde in einer Ablationsstudie untersucht. Jeweils fünf unabhängige Optimierungsläufe mit der Zieltextur-Menge $\mathcal{G}_{2\text{-equiv}}$ wurden mit aktiver *Augmentierung* und ohne *Augmentierung* durchgeführt. Ergebnisse der Studie sind in

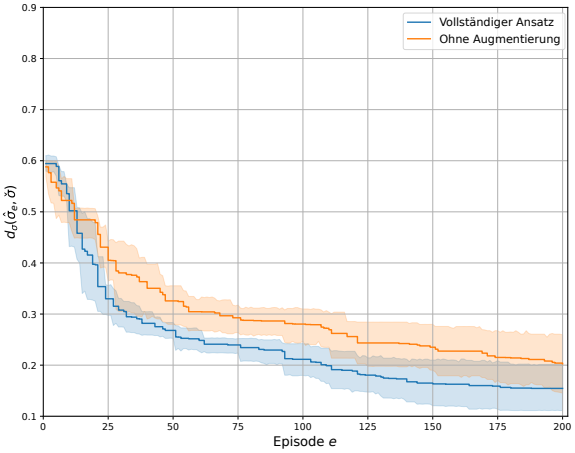



Abbildung 5.16: Ergebnisse der Ablationsstudie für mehrere äquivalente Zielstrukturen. Pro Variante des Algorithmus ist der Verlauf der Distanz $d_\sigma(\hat{\sigma}, \check{\sigma})$ in Form des Mittelwertes und des 95% Konfidenzintervalls für fünf unabhängige Optimierungsläufe dargestellt.

Abbildung 5.16 dargestellt. Wie schon bei der Evaluierung der Methode für einzelne Zieltexturen wurde als Qualitätsmaß die Distanz der besten bisherigen Lösung in Abhängigkeit von $e$ herangezogen und jeweils der Mittelwert und das 95%-Konfidenzintervall dargestellt. Der Abbildung 5.16 folgend hat die *Augmentierung* einen positiven Einfluss auf die Konvergenzgeschwindigkeit des Algorithmus.

## 5.5 Diskussion der Ergebnisse

Im Folgenden werden die in 5.4.1 vorgestellten Ergebnisse zur Struktur-geleiteten Prozesspfadoptimierung für eine einzelne Zielstruktur und die in 5.4.2 vorgestellten Ergebnisse für die Optimierung mit mehreren äquivalenten Zielstrukturen zusammenfassend bewertet. Eine Diskussion der Algorithmen und Ergebnisse im Gesamtkontext der Arbeit findet in Kapitel 6.1 statt.

In 5.4.1 werden Ergebnisse des SG-SGPPO Algorithmus, angewandt auf den in 5.3 eingeführten generellen Deformationsprozess, für verschiedene einzelne Zielstrukturen vorgestellt. Anhand der quantitativen Darstellung des Distanzmaßes zwischen erreichter Textur und Zieltextur, sowie der qualitativen Darstellung anhand von Polfiguren der erreichten Texturen fällt auf, dass für einige Zieltexturen Prozesspfade gefunden werden können, die sehr nah an die Zieltextur führen, während dies bei anderen Zieltexturen nicht der Fall ist. Solche Zieltexturen, für die SG-SGPPO keinen zufriedenstellenden Prozesspfad ermitteln kann, sind insbesondere $\breve{\sigma}^{(2)}$ und $\breve{\sigma}^{(4)}$. Da, wie beschrieben, häufig mehrere unterschiedliche Strukturen die gewünschten Materialeigenschaften aufweisen, bietet sich an, statt einzelner Zielstrukturen eine Menge bezüglich der Eigenschaften äquivalenter Zielstrukturen vorzugeben. Zur Untersuchung des MEG-SGPPO Algorithmus wurde daher für diese beiden Texturen jeweils eine Menge von 10 Texturen ermittelt, die bezüglich der Eigenschaften als äquivalent betrachtet werden. Wie in 5.4.2 gezeigt, ist es durch MEG-SGPPO möglich, auf dateneffiziente Art gute Prozesspfade für eine oder mehrere dieser äquivalenten Texturen zu ermitteln. So werden im Fall der $\breve{\sigma}^{(4)}$-äquivalenten Texturen bereits während der ersten 25 Episoden bessere Ergebnisse als durch SG-SGPPO ermittelt. Im Fall der $\breve{\sigma}^{(4)}$-äquivalenten Texturen werden während der ersten 75 Episoden bessere Ergebnisse ermittelt.

Wie die exemplarischen Visualisierungen der Prozesspfade, sowohl für SG-SGPPO (Abbildungen 5.8 und 5.11) als auch für MEG-SGPPO (Abbildung 5.15) zeigen, reichen für einige Zieltexturen einfache Prozesspfade, aus nur wenigen unterschiedlichen Aktionen bestehend, aus, um gute Ergebnisse zu

erzielen, während in anderen Fällen Prozesspfade mit deutlich höherer Variation der Aktionen ermittelt werden.
Durch Ablationsstudien konnte nachgewiesen werden, dass einige zentrale Bestandteile der entwickelten Methoden von großer Bedeutung für die Konvergenz der Methoden sind. Abbildung 5.12 zeigt für SG-SGPPO, dass die Umformung der Belohnungsfunktion einen enormen Einfluss auf das Konvergenzverhalten hat. Ohne Umformung der Belohnungsfunktion (grün dargestellt) ist bei den Optimierungsläufen im Verlauf der 98 Episoden nur eine geringfügige Verbesserung des Optimierungsergebnisses feststellbar. Dieser positive Effekt der Umformung ist darauf zurückzuführen, dass durch die Belohnungssignale ungleich null während der Episode die korrekte Identifikation von Aktionen, die zu später beobachteten Belohnungssignalen führen und damit das Lernen der Strategie deutlich erleichtert wird (siehe 2.3.4). Obwohl dieses Experiment nicht für MEG-SGPPO wiederholt wurde, ist zu erwarten, dass die Bedeutung der Umformung vergleichbar hoch ist, da MEG-SGPPO eine Erweiterung von SG-SGPPO darstellt und darüber hinaus bei MEG-SGPPO auch die $\varepsilon$-*greedy* Wahl der Zielstruktur direkt von der Bewertungsfunktion abhängt. Abbildung 5.16 folgend ist für MEG-SGPPO ein positiver Einfluss der Augmentierung durch hypothetische Zieltexturen (siehe 5.2.2) auf die Konvergenzgeschwindigkeit und damit auf die Dateneffizienz des Algorithmus erkennbar. Die verwendeten DQN Erweiterungen (*Prioritized Experience Replay*, *Double Q-Learning* und *Dueling Q-Learning*) haben, Abbildung 5.12 folgend, hingegen keinen bedeutenden Einfluss auf die Konvergenz. Die geringere Breite des Konfidenzintervalls bei gleichem Stichprobenumfang (jeweils fünf unabhängige Optimierungsläufe) deutet jedoch an, dass durch die Erweiterungen eine erhöhte Stabilität des Algorithmus erzielt wird.

# 6 Übergreifende Diskussion und Ausblick

In dieser Arbeit wurden Methoden des modellfreien, bewertungsbasierten bestärkenden Lernens entwickelt und anhand von Fertigungsprozessen mit *endlichem Zeithorizont* für die Ermittlung optimaler Regelungsstrategien untersucht. Dabei wurden zwei grundlegende Problemklassen definiert und spezialisierte Algorithmen zur Lösung entwickelt. Ergänzend zu der Diskussion der Ergebnisse in 4.6 und 5.5, wird im Folgenden der dabei gemachte methodische Beitrag und Untersuchungsergebnisse erläutert und diskutiert. Außerdem werden Anknüpfungspunkte für weitergehende Arbeiten dargestellt. In Abschnitt 6.2 werden anschließend Möglichkeiten skizziert, um die beiden in der Arbeit getrennt behandelten Problemklassen gemeinsam zu behandeln.

## 6.1 Methodische Beiträge und Untersuchungsergebnisse

Eine zentrale Eigenschaft der entwickelten und untersuchten Methoden ist, dass sie ohne Modell des Prozesses optimale Regelungsstrategien durch Interaktion lernen. Diese Eigenschaft unterscheidet die Methoden wesentlich von traditionellen, modellbasierten Methoden der optimalen Regelung, insbesondere der *modellprädiktiven Regelung*. Modellfreie Verfahren sind nicht abhängig von einem Prozessmodell, das häufig entweder den Prozess zu sehr vereinfacht darstellt, oder zu komplex ist, um in Echtzeit während der Prozessausführung

verwendet werden zu können. Des Weiteren ermöglichen die modellfreien Verfahren den Entwurf einer Regelstrategie-Optimierung, ohne über tiefgreifendes Expertenwissen bezüglich des spezifischen Prozesses zu verfügen.
Bei der *modellprädiktiven Regelung* wird das Prozessmodell bei der Ausführung in jedem Zeitschritt zur Planung der nächsten Regelungsaktionen verwendet. Dies führt zu der oben angesprochenen Abwägung zwischen Modellgenauigkeit und Echtzeitfähigkeit. Im Gegensatz dazu ist bei den entwickelten Methoden des bestärkenden Lernens der Aufwand pro Zeitschritt auf die Abfrage des $Q$-Netzes begrenzt, sodass Echtzeitfähigkeit kein Problem darstellt. Das Lernen der Netze geschieht bei den in dieser Arbeit entwickelten Methoden asynchron, durch *Experience Replay* (siehe 2.2.7) und kann parallel zu der optimalen Regelung durchgeführt werden.
In Kapitel 4 wird ein im Rahmen der Arbeit entwickelter modellfreier, approximativer, *bewertungsbasierter* Algorithmus des bestärkenden Lernens, *Backward Fixed-Horizon Neural Q-Learning* (BFHNQ), eingeführt. Dieser basiert, wie *Approximate Backward Dynamic Programming* (ABDP) (Kapitel 2.2.4), auf Updates Zeitschritt-abhängiger Funktionsapproximationen rückwärts in den Zeitschritten. BFHNQ ist ein approximativer Q-Learning Ansatz, der durch einen angepassten, inkrementellen *Neural Fitted Q-Iteration* (NFQ) Mechanismus (Kapitel 2.2.7) Zeitschritt-abhängige Q-Funktionen in nur einer Iteration lernt. Das Prinzip der Zeitschritt-abhängigen Funktionsapproximationen wird durch BFHNQ außerdem genutzt, um dem Problem der partiellen Beobachtbarkeit des Fertigungsprozesses (2.3.1) mit einer Zeitschritt-abhängigen Zustandsbeschreibung zu begegnen.
BFHNQ ist im Ergebnis ein Algorithmus, der wie NFQ in der inkrementellen Variante bei jedem Trainings-Vorgang das gesamte *Replay Memory* zum Training der $Q_t$-Netze nutzt und mit *Batch* Optimierungsverfahren (Siehe Anhang A.0.2) kombinierbar ist. Durch das Training rückwärts in den Zeitschritten ist, entgegen NFQ, nur ein einziges Training der $Q$-Netze notwendig. Das Nutzen

des gesamten Datensatzes bei jedem Training führt zu dateneffizientem Lernen, während der Trainingsaufwand durch das Training rückwärts in den Zeitschritten erheblich gesenkt wird. Wie ABDP ist BFHNQ ein äußerst effizienter Algorithmus. Da pro Zeitschritt ein separates $\mathcal{Q}_t$-Netz gelernt wird, sind beide Algorithmen für Entscheidungsprobleme mit wenigen Zeitschritten entwickelt und nicht für den Einsatz bei Entscheidungsproblemen mit weitem Zeithorizont gedacht. Der wesentliche Unterschied von BFHNQ gegenüber ABDP ist, dass BFHNQ eine Q-Funktionen lernt und somit für *online* Anwendungsfälle entwickelt wurde, während ABDP unter anderrem ein explizites Zustandsübergangsmodell nutzt, um Zustands-Bewertungsfunktionen zu lernen und damit eher für *offline* Anwendungsfälle ausgelegt ist.

Durch das *online*-Lernen der optimalen Regelungsstrategien, adaptieren die entwickelten Verfahren Eigenheiten und Bedingungen des spezifischen Prozesses. Dies wurde für den Fall des simulierten Tiefziehprozesses in Kapitel 4 untersucht, wobei davon ausgegangen wurde, dass der Reibungskoeffizient des Prozesses variabel und nicht direkt beobachtbar ist und das hypothetische modellbasierte Alternativverfahren über ein Prozessmodell verfügt, bei dem von einem statischen Reibungskoeffizienten ausgegangen wird. Wie in den Ergebnissen gezeigt wird (siehe 4.4.1), adaptiert BFHNQ den aktuellen Reibungskoeffizienten und übertrifft so die Ergebnisse der hypothetischen modellbasierten Alternative. Weitergehende Untersuchungen mit unterschiedlichen Beobachtbarkeits-Szenarien zeigen, dass BFHNQ im Szenario ohne beobachtbare Werte im Erwartungswert zu ähnlichen Ergebnissen führt wie das hypothetische Alternativverfahren, das auch unabhängig von den aktuellen Prozessbedingungen operiert. Daneben konnte für den Anwendungsfall gezeigt werden, dass BFHNQ robust bezüglich der verwendeten Q-Learning Parameter $(\alpha, \varepsilon_0)$ ist, was in der Praxis die Instanziierung für neue Prozesse erheblich vereinfacht.

Von zentraler Bedeutung bei der *online*-Optimierung von Fertigungs-Prozessen ist die Dateneffizienz, da eine geringe Dateneffizienz gleichbedeutend mit einer erhöhten Ausschussrate ist. Für einen deterministischen Tiefziehprozess

mit gleichbleibendem Reibungskoeffizienten wurde außerdem die Dateneffizienz von BFHNQ untersucht und mit einem *Hill-Climbing* Ansatz verglichen. Den in 4.4.2 geschilderten Untersuchungsergebnissen folgend ist BFHNQ im Vergleich mit dem einfachen *Hill-Climbing* Ansatz um nahezu eine Größenordnung Effizienter. Aufgrund der zentralen Bedeutung der Dateneffizienz ist dies ein Anknüpfungspunkt für die vergleichende Untersuchung weiterer modellfreier Ansätze.
Eine Möglichkeit, die Dateneffizienz im Fall sich ändernder Rahmenbedingungen und damit verbundener Vorgaben (Konfigurationen) der Entscheidungsoptimierung zu erhöhen, ist die Erweiterung des BFHNQ Algorithmus zu einem multikriteriellen Verfahren. Ziel ist der Transfer von gelernten Bewertungsfunktionen zwischen Konfigurationen, so dass bei einer Änderung der Rahmenbedingungen (und damit verbundenen Optimierungszielen) auf bereits gelerntes Prozesswissen zurückgegriffen werden kann. Ein erster Ansatz für diese Weiterentwicklung wurde in der Arbeit vorgestellt und anhand des Tiefzieh-Optimierungsproblems untersucht. Dabei konnte für den Prozess gezeigt werden, dass ein Transfer des gelernten Prozesswissens stattfindet und so bei einer neu-Definition der Zielstellung bereits ein recht guter Ausgangspunkt für eine effiziente Adaption der neu-konfigurierten Belohnungsfunktion gefunden wurde. Allerdings hat sich bei der Untersuchung der Weiterentwicklung auch herausgestellt, dass BFHNQ ohne die multikriterielle Weiterentwicklung nach einigen hundert Episoden zu besseren Ergebnissen führt. Dies ist auf zwei konkrete Punkte zurück zu führen, die bereits in 4.5 benannt wurden:

1. Die Verteilung der Daten im *Replay-Memory* widerspricht der Annahme des *on-policy* Updates, dass Aktionen der aktuellen Strategie folgend verteilt sind.

2. Die Anwendung nicht-linearer Skalarisierungsfunktionen $\mathfrak{f}$ bei den Untersuchungen widerspricht einer grundlegenden Annahme des multikriteriellen Ansatzes (4.15).

Zur Korrektur des ersten Problems können *Weighted Importance Sampling* oder *Importance Resampling* Methoden [111] verwendet werden, die dazu führen, dass die während des Trainings der Netze die Daten so gewichtet beziehungsweise so verteilt sind, dass sie die aktuelle Strategie besser widerspiegeln. Der gängige Weg, das zweite Problem zu vermeiden, ist die Verwendung einer linearen Skalarisierungsfunktion. Wie in dem Abschnitt 4.5 allerdings gezeigt und erläutert wird, liegen die *Pareto-Optima* auf einer streckenweise linearen *Pareto-Front* und können so nur durch eine konkave Skalarisierungsfunktion ermittelt werden. Eine ideale Lösung wäre deshalb eine Lösung, die ohne die Annahme aus (4.15) auskommt und mit der Verwendung nicht-linearer Skalarisierungsfunktionen vereinbar ist. Eine solche alternative Lösung könnte erreicht werden, wenn abweichend zu der in 4.5 vorgestellten Lösung anstelle der multikriteriellen vektorwertigen Bewertungsfunktion eine generalisierte skalare Bewertungsfunktion $\mathcal{Q}(\mathbf{s}, a, \mathbf{w}, \boldsymbol{\theta}) \approx Q^*_{\mathbf{w}}(s,a)$ gelernt würde, wobei $Q^*_{\mathbf{w}}(s,a)$ die optimale Q-Funktion für das mit $\mathbf{w}$ konfigurierte Belohnungssignal $R = \mathfrak{f}(\mathfrak{r}, \mathbf{w})$ darstellt. Diesem Schema folgt der *Multi-Objective Fitted Q-Iteration* (MOFQ) Ansatz [112]. Es wurden ausschließlich veröffentlichte Arbeiten gefunden, bei denen MOFQ und davon abgeleitete Ansätze für lineare Skalarisierungsfunktionen untersucht wurde. Weitergehende Untersuchungen von MOFQ unter Verwendung konkaver Skalarisierungsfunktionen im Kontext einer linearen oder konkaven *Pareto-Front* sind aufgrund des in dieser Arbeit erkennbar gewordenen Bedarfs von hohem Interesse.

Ein weiteres Anwendungsfeld, das im Rahmen der Arbeit untersucht wird, ist die *Struktur-geleitete Optimierung* von Fertigungsprozessen. Der wesentliche Unterschied zu den vorangegangenen Untersuchungen partiell beobachtbarer Fertigungsprozesse ist, dass das Ziel der Entscheidungsoptimierung die Erreichung bestimmter Material-Strukturen ist und eine Beschreibung der aktuellen Material-Struktur in jedem Zeitschritt als gegeben angenommen wird. Zusammen mit Methoden zur Abbildung von Materialeigenschaften auf Material-Strukturen (siehe 2.4.4) stellt die *Struktur-geleitete Optimierung* von Fertigungsprozessen einen Ansatz zur zielgerichteten Entwicklung von Prozessen

zur Herstellung neuer Materialien mit bestimmten Eigenschaften dar. Die Verwendung von modellfreiem bestärkendem Lernen für *Struktur-geleitete Optimierung* zeichnet sich gegenüber existierenden Ansätzen (siehe 3) dadurch aus, dass *online* gelernt wird und so die Qualität der Optimierungsergebnisse nicht von vorberechneten Datenbanken oder Modellen abhängt. Darüber hinaus sind die in dieser Arbeit entwickelten Methoden in der Lage, auch sehr lange Prozesspfade insbesondere auch stochastischer Fertigungsprozesse zu optimieren. Da bezüglich der gewünschten Materialeigenschaften häufig mehrere äquivalente Material-Strukturen existieren, wurde in dieser Arbeit ein neuartiger Mechanismus des bestärkenden Lernens entwickelt, mit dem auf dateneffiziente Art ein optimaler Prozesspfad zur annähernden Erreichung einer der am besten zu erreichenden äquivalenten Strukturen ermittelt wird. Ein weiteres Alleinstellungsmerkmal der entwickelten Methoden im Kontext der *Struktur-geleiteten Optimierung* ist, dass sie als Methoden des modellfreien bestärkenden Lernens nicht nur zur beschriebenen Ermittlung von Prozesspfaden im Vorfeld der Prozessausführung genutzt werden können, sondern direkt in die optimale Regelung des Fertigungsprozesses integriert werden können. Dieser Aspekt wird in Abschnitt 6.2 gesondert behandelt. Die dergestalt entwickelten Methoden wurden anhand eines verallgemeinerten Metallverarbeitungsprozess untersucht.

Der grundlegende, in dieser Arbeit vorgeschlagene Ansatz zur *Struktur-geleiteten Optimierung* (SG-SGPPO) verbindet den *Deep Q Networks* DQN Algorithmus mit einem für die Problemklasse entwickelten *Potential Based Reward Shaping* Ansatz zur Umformung der Belohnungsfunktion. Durch die Verwendung von *Deep Q Networks* ist es möglich, auch für sehr lange Prozesspfade effizient zu lernen. Die Umformung der Belohnungsfunktion ermöglicht es, den Grad der Erreichung der Zielstruktur erst im letzten Zeitschritt, nach der Prozessausführung, zu bewerten, aber trotzdem effizientes Lernen zu ermöglichen. Wie in den Ergebnissen einer Ablationsstudie für den verallgemeinerten Deformationsprozess gezeigt wurde, ist dies eine wichtige Voraussetzung für die schnelle Konvergenz des Algorithmus.

Zur *Struktur-geleiteten Optimierung* mit mehreren äquivalenten Zielstrukturen wurde ein erweiterter Ansatz entwickelt und vorgestellt: *Multi-Equivalent-Goal Structure-Guided Processing Path Optimization* (MEG-SGPPO). Dieser basiert auf dem grundlegenden *Single-Goal* Ansatz (SG-SGPPO) und erweitert diesen durch einen Mechanismus zur Priorisierung von Zielmikrostrukturen anhand der gelernten Q-Funktion und durch einen Mechanismus zur Augmentierung des *Replay Memory*. Die Priorisierung ermöglicht es, während der Optimierung der Prozesspfade zunehmend den Fokus auf Ziel-Mikrostrukturen zu legen, für die bereits gute Pfade ermittelt wurden. Zu diesem Zweck werden Bewertungsfunktionen gelernt, die den in 2.3.3 eingeführten Ansätzen folgend über Ziele generalisieren.
Der SG-SGPPO Algorithmus wurde für unterschiedliche Zielmikrostrukturen separat evaluiert. Den Ergebnissen ist zu entnehmen, dass für einige Zielmikrostrukturen sehr einfache Prozesspfade identifiziert werden können, während für andere hingegen kompliziertere Prozesspfade mit hoher Variation der ausgewählten Aktionen als optimale Lösung gefunden werden. Wie oben beschrieben, hat die Umformung der Belohnungsfunktion einen sehr hohen Einfluss auf die Konvergenzgeschwindigkeit, während die verwendeten DQN-Erweiterungen (*Prioritized Experience Replay*, *Double Q-Learning* und *Dueling Q-Learning*) bei dem Anwendungsfall keinen hohen Einfluss auf die Ergebnisse haben. Für einige der Zielstrukturen konnte jedoch kein Prozesspfad gefunden werden, welcher in deren Nähe führt.
Zur Untersuchung des MEG-SGPPO Ansatzes wurden für zwei Zielstrukturen, die durch SG-SGPPO nicht gut erreicht werden konnten, jeweils Mengen von 10, bezüglich der Eigenschaften (in diesem Fall der Elastizitätsmoduli) äquivalente, Zielstrukturen ermittelt. MEG-SGPPO wurde auf die beiden Mengen separat angewandt. Wie erhofft konnte der Algorithmus in beiden Fällen erreichbare äquivalente Zielstrukturen identifizieren und Prozesspfade finden, die nahe an die identifizierten Zielstrukturen führen.

Zur Durchführung der Untersuchungen wurde eine generalisierte Experimentalumgebung implementiert. Im Gegensatz zu anderen Bereichen des maschinellen Lernens findet bestärkendes Lernen nicht mittels vorerzeugter Daten, sondern in einem Prozess der Interaktion mit der Daten-erzeugenden *Umgebung* statt. Die durchgeführten Untersuchungen zeichnen sich gegenüber einem Großteil der verwandten Arbeiten durch hohe rechnerische Kosten zur Darstellung der Umgebung durch physikalische Simulation aus. Insbesondere die Simulation des Tiefziehprozesses ist trotz des reduzierten Geometriemodells rechnerisch aufwändig. Zentrale Zielstellungen der in 4.3.5 geschilderten Softwarearchitektur, neben der Reproduzierbarkeit, ist deshalb die Fähigkeit, Experimente parallel durchzuführen und die Wiederverwertbarkeit von Simulationsergebnissen sicher zu stellen. Durch die Unterstützung der *OpenAI Gymar* Schnittstelle wird die Instanziierung neuer Prozesssimulationen und die Evaluation neuer Methoden des bestärkenden Lernens vereinfacht.
Die wichtigsten methodischen Beiträge der Arbeit sind im Folgenden zusammenfassend aufgelistet:

1. Die Entwicklung von BFHNQ als dateneffizienter Algorithmus zur modellfreien Lösung partiell beobachtbarer Entscheidungsprobleme mit festem Zeithorizont.

2. Die Entwicklung einer multikriteriellen Erweiterung des BFHNQ Algorithmus zum Umgang mit sich ändernden Gewichtungen der Optimierungskriterien.

3. Die Beschreibung der Prozessoptimierung im Kontext der Inversen Optimierung der Kausalitätskette Prozess, Material-Struktur, Material-Eigenschaften als Markov Entscheidungsprozess und die Entwicklung eines Ansatzes zur modellfreien Lösung.

4. Die Definition von Markov Entscheidungsprozessen mit mehreren äquivalenten Zielen und die Entwicklung eines Ansatzes zum dateneffizienten Umgang mit derartigen Problemen.

5. Die Entwicklung einer $\chi^2$-Distanzfunktion für kristallographische Texturen.

Diese methodischen Erweiterungen sind zentrale Bausteine einer intelligenten, autonom lernenden Fertigung. Diese lernt unter der Vorgabe der Zieleigenschaften des Produkts eine adaptive Strategie zur Erreichung der für das Ziel erforderlichen Bauteileigenschaften und Material-Struktur. Mögliche Anknüpfungspunkte weiterer Forschung zur Erreichung dieses übergreifenden Ziels werden im folgenden Abschnitt geschildert.

## 6.2 Zusammenführung und Erweiterung der entwickelten Methoden

Die Problemklassen *partiell-beobachtbare Fertigungsprozesse unter stochastischen Einflüssen* und *Struktur-geleitete Optimierung von Fertigungsprozessen* werden in der Arbeit weitgehend getrennt voneinander in den Kapiteln 4 und 5 behandelt. Während in Kapitel 4 die *online* Anwendbarkeit auf Fertigungsprozesse mit damit verbundenen Besonderheiten wie partieller Beobachtbarkeit und stochastischen Prozessbedingungen im Mittelpunkt steht, liegt in Kapitel 5 der Fokus auf den Besonderheiten bei der Definition des Optimierungsziels im Raum der Material-Strukturen.
Neben Einzelprozessen sind die in dieser Arbeit vorgestellten Methoden durch geringfügige Anpassungen auch auf Prozessketten anwendbar. Voraussetzung hierfür ist, dass ein gemeinsamer Zustandsraum $S$ (beziehungsweise eine gemeinsame Form der Zustandsbeschreibung $\mathbf{s}$ oder pseudo-Zustandsbeschreibung $\tilde{\mathbf{s}}$) gefunden werden kann, der es erlaubt, alle Prozesse der Kette als Markov-Entscheidungsprozess zu formulieren. Im Fall der Struktur-geleiteten Optimierung ist dies ohne weiteres gegeben, da die Zustandsbeschreibung im Wesentlichen aus der Beschreibung der aktuellen Material-Struktur besteht.

Wie eingangs in 2.2.1 besprochen, erlaubt die Definition als Markov-Entscheidungsprozess die Nutzung Zustands-abhängiger Aktionsräume $A_s$. Die *Bellman-Optimalitätsgleichungen* und darauf basierende bewertungsbasierte Algorithmen sind auf diesen Fall direkt übertragbar und auch approximative Algorithmen können durch geringfügige Anpassungen weiterverwendet werden (siehe beispielsweise [32]). Diese Flexibilität ermöglicht die Anwendung der Methoden zur Optimierung vordefinierter Ketten von unterschiedlichen Fertigungsprozessen (beispielsweise Wärmebehandlung und Deformation von Stahl). Darüber hinaus kann auch die Reihenfolge der Prozessausführungen selbst als Teil des Entscheidungsproblems gesehen, und durch den Agenten optimiert werden.

Obwohl die in Kapitel 5 entwickelten Algorithmen zur *Struktur-geleiteten Optimierung von Fertigungsprozessen* (SGPPO-Algorithmen) im Rahmen der Arbeit anhand eines deterministischen Prozesses untersucht wurden, sind sie grundsätzlich auch auf stochastische Entscheidungsprozesse *online* anwendbar, da alle Entscheidungen auf Basis der Bewertungsfunktionen getroffen werden (insbesondere auch die Wahl der Zielstruktur im Fall des MEG-SGPPO). Eine wichtige Voraussetzung hierbei ist die Beobachtbarkeit der aktuellen Mikrostruktur in jedem Zeitschritt zur Berechnung der umgeformten Belohnung und als Grundlage der Zustandsbeschreibung. Dies ist bei der *online*-Optimierung, wie sie in Kapitel 4 behandelt wurde, häufig nicht der Fall. Anders als bei den in Kapitel 4 entwickelten Methoden muss für die SGPPO-Algorithmen allerdings die explizite Beschreibung der Material-Struktur als Zustandsbeschreibung gegeben sein und eine künstliche Beschreibung, für die die *Markov-Annahme* getroffen wird, reicht nicht aus. Eine modellbasierte Möglichkeit, die explizite Beschreibung im partiell beobachtbaren Fall zu approximieren, ist die Nutzung eines gelernten oder explizit modellierten Beobachtungsmodells als Punktschätzer der aktuellen Material-Struktur (siehe 2.3.1).

# Literaturverzeichnis


[1] M. Senn, K. Jöchen, T. P. Van, T. Böhlke, and N. Link, “In-depth online monitoring of the sheet metal process state derived from multi-scale simulations,” *The International Journal of Advanced Manufacturing Technology*, vol. 68, no. 9, pp. 2625–2636, 2013.

[2] M. Senn, N. Link, J. Pollak, and J. H. Lee, “Reducing the computational effort of optimal process controllers for continuous state spaces by using incremental learning and post-decision state formulations,” *Journal of Process Control*, vol. 24, no. 3, pp. 133–143, 2014.

[3] M. Senn, “Optimale prozessführung mit merkmalsbasierter zustandsverfolgung,” Dissertation, Karlsruher Institut für Technologie, Karlsruhe: KIT Scientific Publishing, 2013.

[4] S. Fischer, O. Hensgen, M. Elshaabiny, and N. Link, “Generating low-dimensional, nonlinear process representations by ordered features,” in *15th IFAC Symposium on Information Control in Manufacturing, Ottawa*. Amsterdam: Elsevier, 2015, pp. 1037–1042.

[5] S. Witt, “Online-zustandstracking mit datengetriebenen prozessmodellen,” Dissertation, Karlsruher Institut für Technologie, 2018.

[6] T. P. Van, K. Jöchen, and T. Böhlke, “Simulation of sheet metal forming incorporating ebsd data,” *Journal of Materials Processing Technology*, vol. 212, no. 12, pp. 2659–2668, 2012.

[7] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. Cambridge, MA: MIT press, 2016.

[8] J. Dornheim, L. Morand, S. Zeitvogel, T. Iraki, N. Link, and D. Helm, "Deep reinforcement learning methods for structure-guided processing path optimization," *Journal of Intelligent Manufacturing*, 2021, [Accepted, In Press].

[9] J. Pagenkopf, "Bestimmung der plastischen anisotropie von blechwerkstoffen durch ortsaufgelöste simulationen auf gefügeebene," Dissertation, Karlsruher Institut für Technologie, Stuttgart: Fraunhofer Verlag, 2019.

[10] A. Schwartz, "A reinforcement learning method for maximizing undiscounted rewards," in *Proceedings of the 10th International Conference on Machine Learning (ICML), Amherst, MA*. San Francisco, CA: Morgan Kaufmann, 1993, pp. 298–305.

[11] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. Hoboken, NJ: John Wiley & Sons, 2014.

[12] R. A. Howard, *Dynamic programming and markov processes*. Cambridge, MA: MIT press, 1960.

[13] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, vol. 6, no. 5, pp. 679–684, 1957.

[14] ——, *Dynamic programming*. Princeton, NJ: Princeton University Press, 1957.

[15] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. Hoboken, NJ: John Wiley & Sons, 2007.

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction, second edition*. Cambridge, MA: MIT press, 2018.

[17] H. S. Nwana *et al.*, “Software agents: An overview,” *Knowledge engineering review*, vol. 11, no. 3, pp. 205–244, 1996.

[18] C. J. C. H. Watkins, “Learning from delayed rewards,” Dissertation, Cambridge: King’s College, 1989.

[19] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[20] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166*. Cambridge: University of Cambridge, Department of Engineering, 1994.

[21] R. S. Sutton, “Generalization in reinforcement learning: Successful examples using sparse coarse coding,” in *Advances in Neural Information Processing Systems 9 (NIPS), Denver, CO*. Cambridge, MA: MIT Press, 1996, pp. 1038–1044.

[22] M. Riedmiller, “Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method,” in *16th. European Conference on Machine Learning (ECML), Porto*. Berlin, Heidelberg: Springer, 2005, pp. 317–328.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529–533, 2015.

[24] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” in *4th International Conference on Learning Representations (ICLR), San Juan*. OpenReview.net, 2016, pp. 1–21.

[25] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the Thirtieth AAAI conference on artificial intelligence (AAAI-16), Phoenix, AZ.* Palo Alto, CA: AAAI Press, 2016, pp. 2094–2100.

[26] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33th International Conference on Machine Learning (ICML), New York City, NY*. MLR.press, 2016, pp. 1995–2003.

[27] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32th International Conference on Machine Learning (ICML), Lille*. MLR.press, 2015, pp. 1889–1897.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347v2*, 2017.

[29] S. Gu, T. Lillicrap, Z. Ghahramani, R. Turner, and S. Levine, "Q-prop: Sample-efficient policy gradient with an off-policy critic," in *5th International Conference on Learning Representations (ICLR), Toulon*. OpenReview.net, 2017, pp. 1–13.

[30] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," in *5th International Conference on Learning Representations (ICLR), Toulon*. OpenReview.net, 2017, pp. 1–20.

[31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning (ICML), Stockholm*. MLR.press, 2018, pp. 1861–1870.

[32] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[33] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proceedings of the 7th International Conference on Machine Learning (ICML), Austin, TX*. San Francisco, CA: Morgan Kaufmann, 1990, pp. 216–224.

[34] D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.

[35] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing Systems 31 (NIPS), Long Beach, CA*. Red Hook: Curran Associates, 2017, pp. 908–918.

[36] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in *Advances in Neural Information Processing Systems 32 (NeurIPS), Vancouver, BC*. Red Hook: Curran Associates, 2019, pp. 12 519–12 530.

[37] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2013.

[38] L.-J. Lin and T. M. Mitchell, "Reinforcement learning with hidden states," in *From Animals to Animats 2: Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: MIT press, 1993, pp. 271–280.

[39] B. Bakker, "Reinforcement learning with long short-term memory," in *Advances in Neural Information Processing Systems 15 (NIPS),*

*Vancouver, BC*, vol. 14. Cambridge, MA: MIT press, 2002, pp. 1475–1482.

[40] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *2015 AAAI Fall Symposium Series, Arlington, VA*. Palo Alto, CA: AAAI Press, 2015, pp. 29–37.

[41] C. Liu, X. Xu, and D. Hu, "Multiobjective reinforcement learning: A comprehensive overview," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 45, no. 3, pp. 385–398, 2015.

[42] S. Natarajan and P. Tadepalli, "Dynamic preferences in multi-criteria reinforcement learning," in *Proceedings of the 22nd International Conference on Machine Learning (ICML), Bonn*. Association for Computing Machinery, 2005, pp. 601–608.

[43] D. C. K. Ngai and N. H. C. Yung, "A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 509–522, 2011.

[44] A. Taylor, I. Dusparic, E. Galvan-Lopez, S. Clarke, and V. Cahill, "Accelerating learning in multi-objective systems through transfer learning," in *International Joint Conference on Neural Networks (IJCNN), Beijing*. IEEE, 2014, pp. 2298–2305.

[45] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup, "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Taipei*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 761–768.

[46] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proceedings of the 32th International*

*Conference on Machine Learning (ICML), Lille*. MLR.press (open access), 2015, pp. 1312–1320.

[47] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems 31 (NIPS), Long Beach, CA*. Red Hook: Curran Associates, 2017, pp. 5048–5058.

[48] T. Schaul and M. Ring, "Better generalization with forecasts," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI), Beijing*. Palo Alto, CA: AAAI Press, 2013, pp. 1656–1662.

[49] B. Bakker, J. Schmidhuber *et al.*, "Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization," in *Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8), Amsterdam*. IAS Society, 2004, pp. 438–445.

[50] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning (ICML), Sydney*. MLR.press, 2017.

[51] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the 16th International Conference on Machine Learning (ICML), Bled*. San Francisco, CA: Morgan Kaufmann, 1999, pp. 278–287.

[52] E. F. Camacho and C. B. Alba, *Model predictive control*. London: Springer, 2007.

[53] L. Grüne and J. Pannek, *Nonlinear model predictive control*. Basel: Springer International Publishing, 2017.

[54] A. Alessio and A. Bemporad, “A survey on explicit model predictive control,” in *Nonlinear Model Predictive Control*. Berlin, Heidelberg: Springer, 2009, pp. 345–369.

[55] J. Saint-Donat, N. Bhat, and T. J. McAvoy, “Neural net based model predictive control,” *International Journal of Control*, vol. 54, no. 6, pp. 1453–1468, 1991.

[56] B. M. Åkesson and H. T. Toivonen, “A neural network model predictive controller,” *Journal of Process Control*, vol. 16, no. 9, pp. 937–946, 2006.

[57] D. Görges, “Relations between model predictive control and reinforcement learning,” in *IFAC-PapersOnLine (20th IFAC World Congress, Toulouse)*, vol. 50, no. 1. Amsterdam: Elsevier, 2017, pp. 4920–4928.

[58] J. Shin, T. A. Badgwell, K.-H. Liu, and J. H. Lee, “Reinforcement learning–overview of recent progress and implications for process control,” *Computers & Chemical Engineering*, vol. 127, pp. 282–294, 2019.

[59] R. S. Sutton, A. G. Barto, and R. J. Williams, “Reinforcement learning is direct adaptive optimal control,” *IEEE Control Systems*, vol. 12, no. 2, pp. 19–22, 1992.

[60] I. Koryakovskiy, M. Kudruss, R. Babuška, W. Caarls, C. Kirches, K. Mombaur, J. P. Schlöder, and H. Vallery, “Benchmarking model-free and model-based optimal control,” *Robotics and Autonomous Systems*, vol. 92, pp. 81–90, 2017.

[61] I. Koryakovskiy, M. Kudruss, H. Vallery, R. Babuška, and W. Caarls, “Model-plant mismatch compensation using reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2471–2477, 2018.

[62] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2042–2062, 2018.

[63] F. L. Lewis and K. G. Vamvoudakis, "Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 1, pp. 14–25, 2010.

[64] B. Kiumarsi, F. L. Lewis, H. Modares, A. Karimpour, and M.-B. Naghibi-Sistani, "Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics," *Automatica*, vol. 50, no. 4, pp. 1167–1175, 2014.

[65] J. Günther, P. M. Pilarski, G. Helfrich, H. Shen, and K. Diepold, "Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning," *Mechatronics*, vol. 34, pp. 1–11, 2016.

[66] Y. Ma, W. Zhu, M. G. Benton, and J. Romagnoli, "Continuous control of a polymerization system with deep reinforcement learning," *Journal of Process Control*, vol. 75, pp. 40–47, 2019.

[67] S. Liu, Z. Shi, J. Lin, and Z. Li, "Reinforcement learning in free-form stamping of sheet-metals," *Procedia Manufacturing*, vol. 50, pp. 444–449, 2020.

[68] T. Chai, S. J. Qin, and H. Wang, "Optimal operational control for complex industrial processes," *Annual Reviews in Control*, vol. 38, no. 1, pp. 81–92, 2014.

[69] Y. Jiang, J. Fan, T. Chai, and F. L. Lewis, "Dual-rate operational optimal control for flotation industrial process with unknown

operational model," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4587–4599, 2019.

[70] W. Xue, J. Fan, V. G. Lopez, J. Li, Y. Jiang, T. Chai, and F. L. Lewis, "New methods for optimal operational control of industrial processes using reinforcement learning on two time-scales," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3085–3099, 2019.

[71] J. Bakakeu, S. Tolksdorf, J. Bauer, H.-H. Klos, J. Peschke, A. Fehrle, W. Eberlein, J. Bürner, M. Brossog, L. Jahn *et al.*, "An artificial intelligence approach for online optimization of flexible manufacturing systems," in *Applied Mechanics and Materials*, vol. 882. Trans Tech Publications Ltd, 2018, pp. 96–108.

[72] A. Kuhnle, J.-P. Kaiser, F. Theiß, N. Stricker, and G. Lanza, "Designing an adaptive production control system using reinforcement learning," *Journal of Intelligent Manufacturing*, vol. 32, no. 3, pp. 855–876, 2020.

[73] S. Tommerup and B. Endelt, "Experimental verification of a deep drawing tool system for adaptive blank holder pressure distribution," *Journal of Materials Processing Technology*, vol. 212, no. 11, pp. 2529–2540, 2012.

[74] C. P. Singh and G. Agnihotri, "Study of deep drawing process parameters: A review," *International Journal of Scientific and Research Publications*, vol. 5, no. 1, pp. 2250–3153, 2015.

[75] A. Wifi and A. Mosallam, "Some aspects of blank-holder force schemes in deep drawing process," *Journal of Achievements in Materials and Manufacturing Engineering*, vol. 24, no. 1, pp. 315–323, 2007.

[76] B. Endelt, S. Tommerup, and J. Danckert, "A novel feedback control system – controlling the material flow in deep drawing using

distributed blank-holder force," *Journal of Materials Processing Technology*, vol. 213, no. 1, pp. 36–50, 2013.

[77] B. Endelt, "Design strategy for optimal iterative learning control applied on a deep drawing process," *The International Journal of Advanced Manufacturing Technology*, vol. 88, pp. 3–18, 2017.

[78] P. Fischer, J. Heingärtner, S. Duncan, and P. Hora, "On part-to-part feedback optimal control in deep drawing," *Journal of Manufacturing Processes*, vol. 50, pp. 403–411, 2020.

[79] P. Guo and J. Yu, "Optimal control of blank holder force based on deep reinforcement learning," in *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Macao*. IEEE, 2019, pp. 1466–1470.

[80] J. Dornheim, N. Link, and P. Gumbsch, "Model-free adaptive optimal control of episodic fixed-horizon manufacturing processes using reinforcement learning," *International Journal of Control, Automation and Systems*, vol. 18, no. 6, pp. 1593–1604, 2020.

[81] R. Liu, A. Kumar, Z. Chen, A. Agrawal, V. Sundararaghavan, and A. Choudhary, "A predictive machine learning approach for microstructure optimization and materials design," *Scientific reports*, vol. 5, no. 1, pp. 1–12, 2015.

[82] A. Paul, P. Acar, W.-k. Liao, A. Choudhary, V. Sundararaghavan, and A. Agrawal, "Microstructure optimization with constrained design objectives using machine learning-based feedback-aware data-generation," *Computational Materials Science*, vol. 160, pp. 334–351, 2019.

[83] J. B. Shaffer, M. Knezevic, and S. R. Kalidindi, "Building texture evolution networks for deformation processing of polycrystalline FCC

metals using spectral approaches: Applications to process design for targeted performance," *International Journal of Plasticity*, vol. 26, no. 8, pp. 1183–1194, 2010.

[84] D. Li, H. Garmestani, and B. Adams, "A texture evolution model in cubic-orthotropic polycrystalline system," *International Journal of Plasticity*, vol. 21, no. 8, pp. 1591–1617, 2005.

[85] D. Li, H. Garmestani, and S. Ahzi, "Processing path optimization to achieve desired texture in polycrystalline materials," *Acta Materialia*, vol. 55, no. 2, pp. 647–654, 2007.

[86] P. Acar and V. Sundararaghavan, "Linear solution scheme for microstructure design with process constraints," *AIAA Journal*, vol. 54, no. 12, pp. 4022–4031, 2016.

[87] ——, "Reduced-order modeling approach for materials design with a sequence of processes," *AIAA Journal*, vol. 56, no. 12, pp. 5041–5044, 2018.

[88] S. Sundar and V. Sundararaghavan, "Database development and exploration of process–microstructure relationships using variational autoencoders," *Materials Today Communications*, vol. 25, p. 101201, 2020.

[89] A. Tran, J. A. Mitchell, L. Swiler, and T. Wildey, "An active learning high-throughput microstructure calibration framework for solving inverse structure-process problems in materials informatics," *Acta Materialia*, vol. 194, pp. 80–92, 2020.

[90] H. Bunge and C. Esling, "Texture development by plastic deformation," *Scripta Metallurgica*, vol. 18, no. 3, pp. 191–195, 1984.

[91] V. Pong, S. Gu, M. Dalal, and S. Levine, "Temporal difference models: Model-free deep rl for model-based control," in *6th International*

*Conference on Learning Representations (ICLR), Vancouver*. OpenReview.net, 2018, pp. 1–14.

[92] J. Dornheim and N. Link, "Multiobjective reinforcement learning for reconfigurable adaptive optimal control of manufacturing processes," in *International Symposium on Electronics and Telecommunications (ISETC), Timisoara*. IEEE, 2018, pp. 1–5.

[93] J. D. Yoo, S. W. Hwang, and K.-T. Park, "Factors influencing the tensile behavior of a Fe–28Mn–9Al–0.8 C steel," *Materials Science and Engineering: A*, vol. 508, no. 1, pp. 234–240, 2009.

[94] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," *arXiv preprint arXiv:1606.01540*, 2016.

[95] Dassault Systèmes, *Abaqus 6.14 Scripting User's Guide*. Providence, RI: Dassault Systèmes, 2014.

[96] ——, *Abaqus 6.14 Analysis User's Guide Volume II: Analysis*. Providence, RI: Dassault Systèmes, 2014.

[97] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[98] G. B. Olson, "Computational design of hierarchically structured materials," *Science*, vol. 277, no. 5330, pp. 1237–1242, 1997.

[99] M. Grześ, "Reward shaping in episodic reinforcement learning," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS), São Paulo, Brazil*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2017, p. 565–573.

[100] P. Mannion, S. Devlin, K. Mason, J. Duggan, and E. Howley, "Policy invariance under reward transformations for multi-objective reinforcement learning," *Neurocomputing*, vol. 263, pp. 60–73, 2017.

[101] S. R. Kalidindi, C. A. Bronkhorst, and L. Anand, "Crystallographic texture evolution in bulk deformation processing of FCC metals," *Journal of the Mechanics and Physics of Solids*, vol. 40, no. 3, pp. 537–569, 1992.

[102] G. I. Taylor, "Plastic strain in metals," *Journal of the Institute of Metals*, vol. 62, pp. 307–324, 1938.

[103] D. Q. Huynh, "Metrics for 3D rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, 2009.

[104] R. Quey, A. Villani, and C. Maurice, "Nearly uniform sampling of crystal orientations," *Journal of Applied Crystallography*, vol. 51, no. 4, pp. 1162–1173, 2018.

[105] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[106] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018, [Online; Stand 25. Mai 2021].

[107] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[108] R. Quey, P. Dawson, and F. Barbe, “Large-scale 3D random polycrystals for the finite element method: Generation, meshing and remeshing,” *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 17, pp. 1729–1745, 2011.

[109] F. Bachmann, R. Hielscher, and H. Schaeben, “Texture analysis with MTEX – free and open source software toolbox,” in *Solid State Phenomena*, vol. 160. Baech: Trans Tech Publications, 2010, pp. 63–68.

[110] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” in *Advances in Neural Information Processing Systems 30 (NIPS) Deep Learning Symposium, Barcelona*. OpenReview.net, 2016, pp. 1–14.

[111] M. Schlegel, W. Chung, D. Graves, J. Qian, and M. White, “Importance resampling for off-policy prediction,” in *Advances in Neural Information Processing Systems 32 (NeurIPS), Vancouver, BC*. Red Hook: Curran Associates, 2019, pp. 1799–1809.

[112] A. Castelletti, F. Pianosi, and M. Restelli, “Multi-objective fitted Q-iteration: Pareto frontier approximation in one single run,” in *International Conference on Networking, Sensing and Control, Delft*. IEEE, 2011, pp. 260–265.

[113] H.-J. Bunge, *Texture analysis in materials science: mathematical methods*. London: Butterworths, 1982.

[114] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[115] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function,” *Neural Networks*, vol. 6, no. 6, pp. 861–867, 1993.

[116] D. C. Liu and J. Nocedal, “On the limited memory bfgs method for large scale optimization,” *Mathematical Programming*, vol. 45, pp. 503–528, 1989.

[117] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations (ICLR), San Diego, CA*. OpenReview.net, 2015, pp. 1–15.

[118] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, “On optimization methods for deep learning,” in *Proceedings of the 28th International Conference on Machine Learning (ICML), Bellevue, WA*. Madison, WI: Omnipress, 2011.

[119] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.

# Tabellenverzeichnis

# Abkürzungs- und Symbolverzeichnis

**Lateinische Symbole**

| | |
|---|---|
| $A$ | Menge der Aktionen $a \in A$ eines Markov-Entscheidungsprozesses |
| $a, a_t, a'$ | Aktion; $a_t$ Aktion zum Zeitschritt $t$; $a'$ Nachfolgeaktion |
| $\bar{A}, \bar{A}_\pi$ | *Advantage*-Funktion $\bar{A} : S \times A \to \mathbb{R}$ |
| $\mathcal{A}$ | Approximierte *Advantage*-Funktion $\mathcal{A}(s, a, \theta)$ |
| $\bar{A}_g$ | Generalisierte *Advantage*-Funktion $\bar{A}_g : S \times A \times G \to \mathbb{R}$ |
| $\check{\mathcal{A}}$ | Approximierte Generalisierte *Advantage*-Funktion $\check{\mathcal{A}}(\mathbf{s}, \mathbf{a}, \mathbf{g}, \theta)$ |
| $B_J^\Omega$ | $J$ gleichmäßig verteilte Orientierungen im der *Fundamental-region* bzgl. $\Omega$ |
| $c_i$ | Kostenterme |
| $\mathbf{c}^{\min}, \mathbf{c}^{\max}$ | Empirische minimale und maximale Kosten |
| $C_l^{ij}$ | Koeffizienten der generalisierten Kugelflächenfunktionen |
| $D$ | Trainingsdatensatz zum überwachten Lernen, bestehend aus $(\mathbf{x}, \mathbf{y})$-Tupeln |
| $\mathcal{D}$ | *Replay-Memory* |
| $\mathcal{D}_t$ | Zeitschritt-abhängiges *replay-Memory* |
| $d$ | Binärvariable zur Signalisierung des Episoden-Endes |
| $d_\sigma$ | Strukturdistanz $d_\sigma : \Sigma \times \Sigma \to \mathbb{R}_0^+$ |
| $\mathbf{D}^u$ | Matrix der räumlichen Translationen je FEM-Zelle in $u$-Richtung |
| $e$ | Episoden-Index |
| $\mathcal{E}_e$ | Menge der Erfahrungs-Tupel aus Episode $e$ |

| | |
|---|---|
| $\mathcal{E}_e^{(g)}$ | Hypothetische Erfahrungen aus Episode $e$ für das Ziel $g$ |
| $E_{ii}$ | Elastizitätsmodul in $ii$-Richtung bezüglich des Referenzkoordinatensystems |
| $f, h$ | mehrfach verwendete Funktionssymbole mit lokaler Definition |
| $\mathfrak{f}$ | Skalarisierungsfunktion für Multikriterielles bestärkendes Lernen |
| $f_t$ | Deformationsgrad in Zeitschritt $t$ |
| $\mathcal{F}$ | Umformfunktion beim *potential based reward shaping* $\mathcal{F}: S \times S \to \mathbb{R}$ |
| $\tilde{\mathbf{F}}, \hat{\mathbf{F}}$ | Deformationsmatrizen |
| $G$ | Menge genereller Zielobjekte $g \in G$ |
| $g$ | Generelles Zielobjekt |
| $g^*$ | Optimales Zielobjekt bei *multi-equivalent Goal MDPs* |
| $\mathbf{g}$ | Vektorrepräsentation eines Ziels |
| $\mathcal{G}$ | Menge äquivalenter Zielstrukturen |
| $\mathbf{h}_\sigma$ | Orientierungshistogramm der Textur $\sigma$ |
| $H$ | Gewichtetes harmonisches Mittel $H(\mathbf{x}, \mathbf{w}) \in \mathbb{R}$) |
| $\mathbf{H}^v$ | Matrix der Ausdehnungen je FEM-Zelle in $v$-Richtung |
| $i, j$ | mehrfach verwendete iterationssymbole mit lokaler Definition |
| $J$ | Anzahl der Basisorientierungen bei der Orientierungshistogramm-Darstellung |
| $K$ | Letzter Zeitschritt einer Episode |
| $k$ | Glättungsparameter der Orientierungshistogramm-Darstellung |
| $l$ | GSH-Grad |
| $L$ | Maximaler GSH-Grad einer gekürzten GSH-Repräsentation $\zeta_L^\Omega(\sigma)$ |
| $m, n$ | mehrfach verwendete indexsymbole mit lokaler Definition |
| $\mathbf{M}$ | Matrix der Mises-Vergleichsspannungen je FEM-Zelle |

| | |
|---|---|
| $n_e$ | Episodenanzahl |
| $n_\varepsilon$ | Anzahl der Episoden in denen $\varepsilon_f$ angenähert wird |
| $\check{n}_\varepsilon$ | Anzahl der Episoden in denen $\check{\varepsilon}_f$ bei der Zielstrukturwahl angenähert wird |
| $n_\mathcal{Q}$ | Anzahl der Episoden zwischen Trainings der Q-*Netze* bei BFHNQ, Anzahl der Zeitschritte zwischen Trainings bei DQN |
| $\mathcal{N}$ | Normalverteilung $\mathcal{N}(m, \sigma^2)$ |
| $o, o_t$ | Abstraktes Symbol für observable Größen |
| $\mathbf{o}$ | Vektorräpresentation observabler Größen |
| $\mathcal{O}$ | Orientierungsverteilungsfunktion (ODF) $\mathcal{O}(\lambda)$ |
| $O$ | Beobachtungsfunktion |
| $P$ | Zustandsübergangsfunktion $P : S \times A \times S \to [0,1]$ |
| $\mathcal{P}$ | Prozesspfad, Sequenz von Aktionen |
| $\mathcal{P}_g^*$ | Optimaler Prozesspfad für die Zielstruktur $\check{\sigma}_\mathcal{G}^g$ |
| $p_\beta, q_\beta$ | Parameter der $\beta$-Verteilung |
| $Q, Q_\pi$ | Aktions-Bewertungsfunktion $Q : S \times A \to \mathbb{R}$ |
| $Q^*$ | Aktions-Bewertungsfunktion der optimalen Strategie $\pi^*$ |
| $Q_g$ | Generalisierte Aktions-Bewertungsfunktion $Q_g : S \times A \times G \to \mathbb{R}$ |
| $\mathcal{Q}$ | Approximierte Aktions-Bewertungsfunktion $\mathcal{Q}(\mathbf{s}, \mathbf{a}, \theta)$ |
| $\check{\mathcal{Q}}$ | Approximierte Generalisierte Aktions-Bewertungsfunktion $\check{\mathcal{Q}}(\mathbf{s}, \mathbf{a}, \mathbf{g}, \theta)$ |
| $\mathcal{Q}_t$ | Zeitschritt-abhängige Approximation der Q-Funktion |
| $q(\lambda)$ | Darstellung einer Orientierung $\lambda$ als Einheitsquaternion $q(\lambda) = \mathbf{q}$ |
| $\mathbf{q}$ | Elemente eines Einheitsquaternion $\mathbf{q}_t \in \mathbb{R}^4, \|\mathbf{q}_t\|_2 = 1$ |
| $R$ | Belohnungsfunktion |
| $R_g$ | Zielabhängige Belohnungsfunktion |
| $R'$ | Umgeformte Belohnungsfunktion (*potential based reward shaping*) |

| | |
|---|---|
| $R'_g$ | Umgeformte Zielabhängige Belohnungsfunktion (*potential based reward shaping*) |
| $\mathfrak{R}$ | Vektorwertige Belohnungsfunktion $R : S \to \mathbb{R}^n$ |
| $r, r_t$ | Belohnungssignal $r \in \mathbb{R}$ (ggf. in Zeitschritt $t$) |
| $\hat{r}$ | Ertrag: Summe der zukünftigen diskontierten Belohnungssignale |
| $\hat{r}_\mu$ | Mittlerer beobachteter Ertrag |
| $\hat{r}_\sigma$ | Varianz des beobachteten Ertrags |
| $\mathfrak{r}$ | Belohnungsvektor $\mathfrak{r} \in \mathbb{R}^n$ |
| $\mathbf{R}$ | Rotationsmatrix $R \in \mathbb{R}^{3\times3}$ |
| $S$ | Menge der Zustände eines Markov-Entscheidungsprozesses |
| $s, s_t, s'$ | Abstraktes Zustandssymbol; $s_t$ Zustand zum Zeitschritt $t$; $s'$ Nachfolgezustand |
| $\mathbf{s}, \mathbf{s}_t$ | Vektorrepräsentation eines Zustandes $\mathbf{s} \in \mathbb{R}^m$ |
| $S_t$ | Menge der möglichen Zustände eines Markov-Entscheidungsprozesses zum Zeitschritt $t$ |
| $\mathcal{S}_t$ | Repräsentative Menge von Zustandsbeschreibungen $\mathbf{s}_t$ |
| $\bar{S}$ | Menge der Terminal-Zustände eines Markov-Entscheidungsprozesses |
| $\bar{s}$ | Terminal-Zustand |
| $\tilde{\mathbf{s}}$ | Pseudo-Zustandsbeschreibung |
| $\tilde{\mathbf{S}}$ | Pseudo-Zustandsraum |
| $t$ | Zeitschritt $t \in \mathbb{N}_0^+$ |
| $T$ | Finaler Zeitschritt eines MDP mit festem Zeithorizont $T \in \mathbb{N}^+$, Maximaler Zeitschritt eines MDP mit endlichem Zeithorizont $T \in \mathbb{N}^+$ |
| $\therefore$ $T_l^{ij}$ | Generalized Spherical Harmonics |
| $\mathcal{U}$ | Gleichverteilung, $\mathcal{U}(a,b)$: stetig, über dem Intervall $[a,b]$; $\mathcal{U}_\mathcal{M}$: Gleichverteilung über der Menge $\mathcal{M}$ |
| $u_\bullet, v_\bullet$ | Position in $u$-Richtung bzw. $v$-Richtung |

| | |
|---|---|
| $V, V_\pi$ | Zustands-Bewertungsfunktion $V : S \to \mathbb{R}$; in 5.3.3 und Anhang A.0.1: Gesamtvolumen einer Textur |
| $V^*$ | Zustands-Bewertungsfunktion der optimalen Strategie $\pi^*$ |
| $V_g$ | Generalisierte Zustands-Bewertungsfunktion $V_g : S \times G \to \mathbb{R}$ |
| $\mathcal{V}$ | Approximierte Zustands-Bewertungsfunktion $\mathcal{V}(\mathbf{s}, \theta)$ |
| $\check{\mathcal{V}}$ | Approximierte Generalisierte Zustands-Bewertungsfunktion $\check{\mathcal{V}}(\mathbf{s}, \mathbf{g}, \theta)$ |
| $\mathcal{V}_t$ | Zeitschritt-abhängige Approximation der Zustands-Bewertungsfunktion |
| $V(\lambda)$ | Volumen einer Textur mit Orientierung $\lambda$ |
| $\mathbf{w}$ | Gewichte der Belohnungs-Terme $\mathbf{w} \in [0, 1]$ |
| $\mathbf{w}_\lambda$ | Zuweisungsvektor der Kristallorientierung $\lambda$ |
| $\tilde{\mathbf{w}}_\lambda$ | Geglätteter Zuweisungsvektor der Kristallorientierung $\lambda$ |
| $\mathbf{x}, \mathbf{x}_\mathcal{V}, \mathbf{x}_Q$ | Eingangsdaten zum Training einer Funktionsapproximation |
| $\mathbf{y}, \mathbf{y}_\mathcal{V}, \mathbf{y}_Q$ | Ausgangsdaten zum Training einer Funktionsapproximation |

**Griechische Symbole**

| | |
|---|---|
| $\alpha$ | Lernrate bei Algorithmen des bestärkenden Lernens $\alpha \in (0, 1]$ |
| $\alpha_{PER}$ | Parameter des *prioritized experience replay* |
| $\beta_0$ | Parameter des *prioritized experience replay* |
| $\Gamma$ | Menge unterschiedlicher Zielstrukturen |
| $\gamma$ | Diskontierungsfaktor eines Markov-Entscheidungsprozesses $\gamma \in [0, 1]$ |
| $\varepsilon$ | Explorationsrate bei $\varepsilon$-greedy Algorithmen $\varepsilon \in [0, 1]$ |
| $\check{\varepsilon}$ | Explorationsrate bei der Zielstrukturwahl |
| $\varepsilon_0$ | Initiale Explorationsrate |
| $\varepsilon_f$ | Finale Explorationsrate |
| $\varepsilon_\lambda$ | Zerfallsrate von $\varepsilon$ |
| $\check{\varepsilon}_0$ | Initiale Explorationsrate bei der Zielstrukturwahl |
| $\check{\varepsilon}_f$ | Finale Explorationsrate bei der Zielstrukturwahl |
| $\zeta$ | GSH-Repräsentation einer Textur $\zeta_L^\Omega(\sigma)$ |

| | |
|---|---|
| $\eta$ | numerische Repräsentation einer Struktur $\eta : \Sigma \to \mathbb{R}^n$ |
| $\theta, \theta^-, \theta_t$ | Parameter eines künstlichen neuronalen Netzes |
| $\kappa$ | Zieleigenschaft $\kappa \in \mathbb{R}^n$ |
| $\Lambda$ | Repräsentative Stichprobe von Kristallorientierungen $\lambda$ einer ODF |
| $\lambda$ | Allgemeine Orientierung $\lambda \in SO(3)$ |
| $\mu$ | Reibungskoeffizient |
| $\Lambda$ | Menge von Observationen $o \in \Lambda$ |
| $\pi$ | nichtdeterministische Strategie $\pi : S \times A \to [0,1]$ |
| $\bar{\pi}$ | deterministische Strategie $\bar{\pi} : S \to A$ |
| $\tilde{\pi}$ | explorative Lernstrategie |
| $\pi^*, \bar{\pi}^*$ | optimale nichtdeterministische / deterministische Strategie |
| $\pi_g^*$ | Optimale Strategie für die Zielstruktur $\breve{\sigma}_{\mathcal{G}}^g$ |
| $\sigma, \sigma_t$ | Material-Struktur $\sigma \in \Sigma$, Standardabweichung |
| $\breve{\sigma}$ | Zielstruktur |
| $\sigma^*$ | Der Zielstruktur nächste Erreichbare Struktur |
| $\breve{\sigma}_{\mathcal{G}}^*$ | Durch einen Prozess am besten erreichbare Struktur in $\mathcal{G}$ |
| $\breve{\sigma}_{\mathcal{G}}^{\sim}$ | Schätzung der am besten erreichbaren Struktur in $\mathcal{G}$ |
| $\breve{\sigma}_{\mathcal{G}}^{(g)} \in \mathcal{G}$ | $g$ zugehörige Zielstruktur |
| $\hat{\sigma}_e$ | beste bisher gefundene Ergebnis-Textur nach Episode $e$ |
| $\tilde{\sigma}_e \in \Sigma_e$ | Ergebnis-Textur des besten Teilpfades in Episode $e$ |
| $\Sigma$ | Menge der Elemente eines Strukturraumes |
| $\Sigma_{\mathcal{P}}$ | Durch einen Prozess erreichbare Strukturen |
| $\Sigma_e$ | Menge der in Episode $e$ erreichten Strukturen |
| $\Phi$ | Zustands-Potentialfunktion $\Phi : S \to \mathbb{R}$ |
| $\phi$ | Distanzmetrik für Orientierungen $\phi : SO(3) \times SO(3) \to \mathbb{R}_0^+$ |
| $\phi_\Omega$ | Minimale Distanz aller Equivalenten Orientierungen bezüglich des Kristallsystems $\Omega$ $\phi_\Omega : SO(3) \times SO(3) \to \mathbb{R}_0^+$ |
| $\chi^2$ | Distanzmaß für Orientierungshistogramme |
| $\Psi_\Omega$ | $\Psi_\Omega(\lambda)$ bildet auf die Menge aller equivalenten Orientierungen bezüglich des Kristallsystems $\Omega$ ab |

| | |
|---|---|
| $\Omega$ | Kristallsystem |

**Operatoren und Sonstige Symbole**

| | |
|---|---|
| $\|\|\mathbf{M}\|\|_F$ | Frobeniusnorm der Matrix $M$ |
| $\|\|\mathbf{v}\|\|_i$ | $i$-Norm des Vektors $v$ |
| $\mathbb{1}_n$ | Einsvektor $\mathbb{1}_n \in \mathbb{R}^n$ |
| $\nabla$ | Nabla Differentialoperator |
| $R^2$ | Determinationskoeffizient |
| $SO(3)$ | Gruppe der 3D Rotationen |
| $\mathbf{v}_a \frown \mathbf{v}_b$ | Vektor Konkatenation $\bullet \frown \bullet : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^{m+n}$ |

**Abkürzungen**

| | |
|---|---|
| ABDP | *Approximate Backward Dynamic Programming* Algorithmus |
| BFHNQ | *Backward Fixed Horizon Neural Q-Learning* Algorithmus |
| BHF | Niederhaltekraft (*Blank Holder Force*) |
| DQN | *Deep Q-Network* Algorithmus |
| FEM | Methode der Finiten Elemente |
| GSH | Generalized Spherical Harmonics (Generalisierte Kugelflächenfunktionen) |
| i.i.d. | Unabhängig, identisch verteilt (independent, identically distributed) |
| L-BFGS | Limited-Memory Broyden–Fletcher–Goldfarb–Shanno Algorithmus |
| MDP | Markov-Entscheidungsprozess (Markov Decision Process) |
| MEG-MDP | *Markov-Entscheidungsprozess mit mehreren äquivalenten Zielen* (*multi-equivalent goal MDP*) |
| MEG-SGPPO | *Multi-Equivalent-Goal Structure-Guided Processing Path Optimization* Algorithmus |
| MOFQ | *Multi-Objective Fitted Q-Iteration* Algorithmus |
| MPC | Model Predictive Control |
| MSE | Mittlerer Quadratischer Fehler (Mean Squared Error) |

| | |
|---|---|
| NFQ | *Neural Fitted Q-Iteration* Algorithmus |
| ODF | Orientierungsdichteverteilungsfunktion (*Orientation Distribution Function*) |
| ReLU | Rectified Linear Unit |
| SG-SGPPO | *Single-Goal Structure-Guided Processing Path Optimization* Algorithmus |

# A Weitere Grundlagen

## A.0.1 Orientierungsdichteverteilungsfunktionen und Generalized Spherical Harmonics

*Generalized Spherical Harmonics* (GSH) sind generalisierte Kugelflächenfunktionen und werden zur Beschreibung von Orientierungsdichteverteilungsfunktionen (engl. *Orientation Distribution Function*, ODF, [113] Seite 42 ff.) im Spektralraum verwendet. Die Orientierungsdichteverteilungsfunktion ODF $\mathcal{O}(\lambda)$ stellt die empirische Wahrscheinlichkeitsdichte einer Stichprobe von Kristallen im Orientierungsraum $SO(3)$ dar

$$\frac{V(\lambda)}{V} = \mathcal{O}(\lambda)d\lambda \quad \text{mit} \quad \int_{SO(3)} f(\lambda)\mathrm{d}\lambda = 1, \tag{A.1}$$

wobei $V(\lambda)$ das kumulierte Volumen der Kristalle mit Orientierung $\lambda \in SO(3)$ und $V$ das Gesamtvolumen der Stichprobe repräsentiert.
Durch Kristall- und Probensymmetrie existieren äquivalente Regionen in $SO(3)$. Diese können auf eine *Fundamentalregion* abgebildet werden, sodass eine ODF $\mathcal{O}(\tilde{\lambda})$ von auf die Fundamentalregion abgebildeten Orientierungen $\tilde{\lambda}$ von $\mathcal{O}(\lambda)$ physikalisch nicht unterscheidbar ist.
Die symmetrisierte Spektralraumdarstellung

$$\mathcal{O}(\lambda) = \sum_{l=0}^{\infty} \sum_{i=1}^{M(l)} \sum_{j=1}^{N(l)} C_l^{ij} \overset{\therefore}{T}_l^{ij}(\lambda) \tag{A.2}$$

für die Kugelflächenfunktionen $\dddot{T}_l^{ij}$ ermöglicht eine niedrig-dimensionale Repräsentation von $\mathcal{O}(\lambda)$ in Form der Koeffizienten $C_l^{ij}$. Durch die Berücksichtigung der Kristall- und Probensymmetrie wird die Anzahl der Koeffizienten reduziert. $M(l)$ ist dabei die Anzahl der linear unabhängigen Kugelflächenfunktionen $\dddot{T}_l^{ij}$ bezüglich der vorliegenden Kristallsymmetrie und $N(l)$ die Anzahl der linear unabhängigen Kugelflächenfunktionen bezüglich der vorliegenden Probensymmetrie. Eine tiefergehende Einführung der GSH und die Erläuterung der Symmetrie-Spezifischen Auswahl der Kugelflächenfunktionen $\dddot{T}_l^{ij}$ findet sich in dem Textbuch *Texture analysis in materials science: mathematical methods* ([113] Seite 47 ff.).
In Abhängigkeit des Szenarios wird die Reihe (A.2) abgeschnitten und die Koeffizienten werden bis zu einem Grad $L$ berücksichtigt. In der Arbeit repräsentiert $\zeta_L^{\Omega}(\sigma)$ den Vektor der GSH Koeffizienten $C_l^{ij}$ mit $l \leq L$ Für eine Textur $\sigma$ mit dem Kristallsystem $\Omega$.

## A.0.2 Künstliche neuronale feedforward Netze zur Funktionsapproximation

In diesem Kapitel werden tiefe künstliche neuronale *feedforward* Netze wie sie in dieser Arbeit zur Funktionsapproximation verwendet werden behandelt. Abweichende Strukturen von neuronalen Netzen, wie rekurrente neuronale Netze, sowie spezialisierte Netzschichten zur Verarbeitung strukturierter Daten, wie beispielsweise 2D-Faltungsschichten, spielen in dieser Arbeit keine direkte Rolle, weshalb an dieser Stelle auf die ausführliche Darstellung in [7] verwiesen wird.
Künstliche neuronale *feedforward* Netze sind generelle Methoden zur üblicherweise nicht-linearen Approximation von Funktionen $\mathbf{y} = f(\mathbf{x})$. Ein künstliches neuronales *feedforward* Netz stellt eine Abbildung $\hat{\mathbf{y}} = \tilde{f}(\mathbf{x}, \theta)$ dar. Gradientenbasierte Optimierung und der *backpropagation* Algorithmus werden genutzt

um die Parameter $\theta$ aus Daten zu lernen, sodass die Approximation $\tilde{f}$ die, üblicherweise unbekannte, zu approximierende Funktion $f$ annähert.
Ein künstliches neuronales *feedforward* Netz besteht aus mehreren aufeinanderfolgenden Neuronen-Schichten (engl *Layer*). Jede Schicht stellt eine vorerst eigenständige Funktion $\bar{\mathbf{y}} = \tilde{f}^{(i)}(\bar{\mathbf{x}}, \theta_i)$ mit Funktionsparametern $\theta_i$ dar. Der Informationsfluss bei einem *feedforward* Netz läuft gerichtet von der *Eingangsschicht* (hier $\tilde{f}^{(0)}$) durch *versteckte Schichten* (hier $\tilde{f}^{(1)}$) hin zur *Ausgabeschicht* (hier $\tilde{f}^{(2)}$), so dass das *feedforward* Netz auch als geschachtelte Funktion $\tilde{f}(\mathbf{x}, \theta) = \tilde{f}^{(2)}(\tilde{f}^{(1)}(\tilde{f}^{(0)}(\mathbf{x}), \theta_1), \theta_2)$ für $\theta_i \subset \theta$ betrachtet werden kann. Die Anzahl der Schichten macht die „tiefe“ des Netzes aus, die Anzahl der Neuronen pro Schicht die „weite“ des Netzes. Der Begriff *tiefes neuronales Netzwerk* und davon abgeleitete Begriffe wie *tiefes Maschinelles Lernen* (engl. *deep learning*) oder *tiefes bestärkendes Lernen* beziehen sich auf diesen Begriff der „Tiefe“ [7]. In dieser Arbeit werden im Folgenden künstliche neuronale *feedforward* Netze als Standardform angesehen und als *künstliche neuronale Netze* oder kurz als *Netze* bezeichnet. Im Rahmen der Arbeit werden neuronale Netze verwendet um beschränkte reelwertige Funktionen $f : \mathbb{R}^n \to \mathbb{R}^m$ zu approximieren. Die folgende formale Einführung geht von diesem Fall aus.
Die Eingabeschicht entspricht in der hier verwendeten Darstellung der Identitätsabbildung $\tilde{f}^{(0)} : \mathbb{R}^n \to \mathbb{R}^n$. Die versteckte Schicht $\tilde{f}^{(1)} : \mathbb{R}^n \to \mathbb{R}^k$ besteht aus einer linearen Transformation mit anschließender Anwendung einer Nichtlinearität $h$, auch *Aktivierungsfunktion* genannt

$$\bar{\mathbf{y}} = h({\mathbf{W}^{(1)}}^T \bar{\mathbf{x}} + \mathbf{b}^{(1)}). \tag{A.3}$$

wobei $(\mathbf{W}^{(1)}, \mathbf{b}^{(1)}) = \theta_1$. Im hier beschriebenen allgemeinen Fall wird üblicherweise die *rectifying Linear Unit* (*ReLU*) als Nichtlinearität verwendet. Elementweise ist die *ReLU* definiert als $h(z_i) = max[0, z_i]$. Die Ausgabeschicht $\tilde{f}^{(2)} : \mathbb{R}^k \to \mathbb{R}^m$ entspricht im Fall der Regression einer weiteren linearen Transformation

$$\hat{\mathbf{y}} = \mathbf{W}^{(2)^T}\bar{\mathbf{x}} + \mathbf{b}^{(2)}, \tag{A.4}$$

wobei $(\mathbf{W}^{(2)}, \mathbf{b}^{(2)}) = \theta_2$.
Das Beispielnetz zur Regression $\tilde{f}$ ist zusammenfassend definiert durch

$$\hat{\mathbf{y}} = \mathbf{W}^{(2)^T} h(\mathbf{W}^{(1)^T}\bar{\mathbf{x}} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}. \tag{A.5}$$

Weitere versteckte Schichten können zur Erhöhung der Modellkapazität eingefügt werden. Das ursprüngliche *universal approximation theorem* [114] besagt, dass beliebige Borel-messbare Funktionen durch neuronale Netze mit linearer Ausgabefunktion und mindestens einer versteckten Schicht mit einer sogenannten *squashing*-Aktivierungsfunktion durch Verbreiterung der versteckten Schicht beliebig genau repräsentiert werden können. In [115] wurde die Gültigkeit des Theorems auch für Netze mit nicht-polynomialer Aktivierungsfunktion nachgewiesen. Der hier betrachtete Fall beschränkt reelwertiger Funktionen $f : \mathbb{R}^n \to \mathbb{R}^m$ ist Borell-messbar (vgl. [7], Kap. 6.4.1).
Zum Anpassem der Parameter $\theta$ des neuronalen Netzes mittels eines Datensatzes $D$ potentiell verrauschter Stichproben $(\mathbf{x}, \mathbf{y})$, dem sogenannten Training, wird eine Kostenfunktion benötigt und ein gradientenbasierter Optimierungsalgorithmus in Kombination mit dem *backpropagation* Verfahren zur Ableitung der Netzparameter bezüglich einer Kostenfunktion angewandt. Eine Grundannahme beim Training ist, dass Stichproben $(\mathbf{x}, \mathbf{y})$ im Trainingsdatensatz $D$ unabhängig voneinander gezogen wurden und aus einer identischen Verteilung $P_{data}$ stammen (üblicherweise, sowie in der Arbeit als i.i.d. Annahme für *independent, identically distributed*).
Das Lernen von Parametern $\theta$ eines neuronalen Netzes unterscheidet sich von anderen Optimierungsproblemen darin, dass die Kostenfunktion $J(\theta)$ auf Basis der Trainingsdaten $(\mathbf{x}, \mathbf{y}) \in D$ formuliert und minimiert wird, das eigentliche Lernziel aber die Minimierung der erwarteten Kosten bezüglich der Gesamtverteilung $J(\theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim P_{data}}[\bullet]$ ist (vgl. [7], Kap. 6.4.1). Die Minimierung der

auf Basis der Trainingsdaten berechneten Surrogat-Kosten ohne Berücksichtigung des eigentlichen Lernziels kann bei ausreichender Anzahl von Parametern zur Überanpassung (engl. *Overfitting*) des Neuronalen Netzes führen. Dem wird durch Regularisierung des Netzes (siehe [7] Kap. 7) begegnet.
Eine typische Kostenfunktion zum Lernen von Neuronalen Netzen setzt sich aus einem Datenterm und einem oder mehreren Termen zur Regularisierung der Netzparameter zusammen. Eine gängige Wahl des Datenterms zur reellwertigen Funktionsapproximation mittels neuronaler Netze ist der mittlere Quadratische Fehler (*Mean Squared Error*, kurz MSE). Unter der Annahme, dass die Daten einer Normalverteilung $P_{data}(y|x) = \mathcal{N}(f(x), \sigma^2)$ folgen, ist die Minimierung des MSE äquivalent zur *maximum Likelihood* Schätzung der Modellparameter.
Der *Backpropagation* Algorithmus wird beim Training von neuronalen Netzen zur effizienten Berechnung der Gradienten der Kostenfunktion bezüglich der Netzparameter $\nabla_\theta J(\theta)$ genutzt. Hierbei werden die Parameter durch Anwendung der Kettenregel Schicht für Schicht, ausgehend von der Ausgabeschicht, berechnet (siehe [7], Kapitel 6.5).
Deterministische *Batch* Optimierungsverfahren nutzen den gesamten Trainingsdatensatz $D$ bei der Berechnung der Gradienten und zum Update der Parameter in jedem Trainingsschritt. Hierzu muss der gesamte Datensatz zum parameter-Update im Hauptspeicher gehalten werden. Im Gegensatz dazu werden bei stochastischen Verfahren in jedem Trainingsschritt die Paramterer basierend auf einer Teilmenge der Trainingsdaten aktualisiert. Verfahren die nur ein einzelnes Sample zum Update verwenden werden auch *online*-Verfahren genannt. Zum Training tiefer neuronaler Netze wird meist eine dazwischenliegende Strategie angewendet, indem die Updates basierend auf einer Menge aus mehreren Samples, sogenannter *mini-Batches*, durchgeführt werden.
Im Rahmen der vorliegenden Arbeit werden *Limited-Memory Broyden – Fletcher – Goldfarb – Shanno* (L-BFGS) [116] und *Adaptive Moment Estimation* (Adam) [117] zum Training der neuronalen Netze verwendet. L-BFGS ist ein *batch*-Verfahren und der Familie der *quasi-Newton Methoden* zuzurechnen. Es

approximiert die inverse Hesse Matrix ohne die Notwendigkeit diese explizit zu repräsentieren. Für das Lernen von Neuronalen Netzen mit geringer Parameteranzahl mit nur wenigen Trainingsdaten ist L-BFGS häufig anderen Verfahren überlegen [118]. Adam ist ein stochastisches Gradientenabstiegsverfahren, das basierend auf *mini-Batches* und einer adaptiven Schätzung der Momente erster und zweiter Ordnung der Gradienten operiert. Adam wird, neben anderen stochastischen Gradientenabstiegsverfahren, zum Lernen tiefer neuronaler Netze mittels umfangreicher Datensätze verwendet [119].

# B Weitere Abbildungen

## B.0.1 Detaildarstellung der Optimierung im deterministischen Fall

Ergänzend zu der Darstellung der Ergebnisse im deterministischen Fall in 4.4.2, sind in den Abbildungen B.1 und B.2 ein charakteristischer Optimierungslauf des *Hill-Climbing* Ansatzes und ein charakteristischer Lernvorgang des BFHNQ Algorithmus dargestellt. Dargestellt sind die fünf Aktionen $a_0$ bis $a_4$ und die erhaltene Belohnung pro Episode während der Optimierung, mit dem Ziel einen der vier besten Prozesspfade für die gegebene Belohnungsfunktion (siehe Abbildung 4.11) zu ermitteln. Quantitative Ergebnisse für jeweils 100 unabhängige Optimierungsläufe mit derselben Zielstellung sind in Abbildung 4.12, rechts dargestellt. In den fünf abgebildeten Bereichen der Abbildungen B.1 und B.2 werden, pro Episode, die Aktionen $a_0$ bis $a_4$ durch den Farbwert der Punkte kodiert und die am Ende der Episode erhaltene Belohnung durch die vertikale Lage der Punkte kodiert dargestellt.

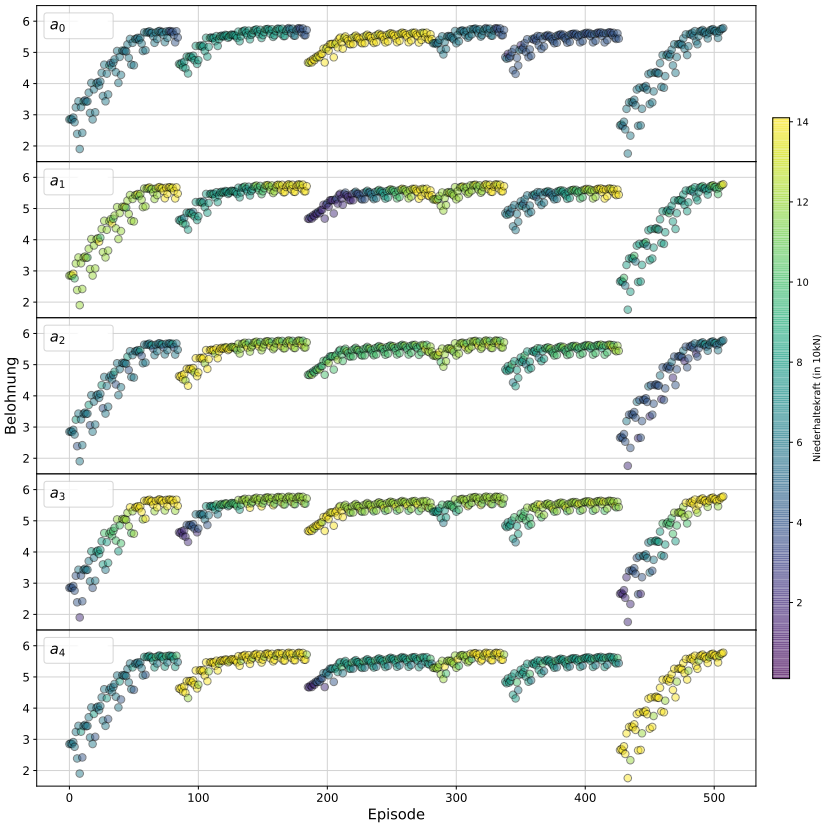



Abbildung B.1: Exemplarischer Optimierungslauf des *Hill-Climbing* Ansatzes als Ergänzung zu den in 4.4.2 dargestellten quantitativen Ergebnissen.

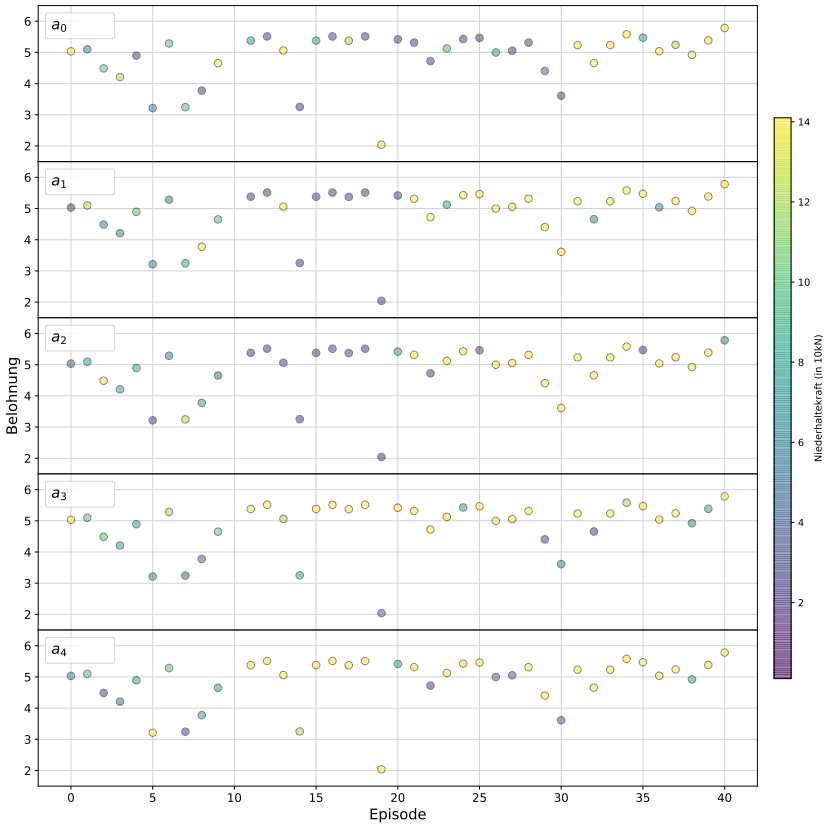



Abbildung B.2: Exemplarischer Optimierungslauf des BFHNQ Algorithmus als Ergänzung zu den in 4.4.2 dargestellten quantitativen Ergebnissen.

## B.0.2 Pareto-Front Abbildungen

In diesem Abschnitt sind, ergänzend zu Abbildung 4.13, Abbildungen der Höhe der Belohnungsterme $\mathfrak{r}_{\text{verbrauch}}$ und $\mathfrak{r}_{\text{wand}}$ für weitere Reibungskoeffizienten $\mu$ dargestellt. Anders als im Fall $\mu = 0.028$ Basieren stellen die Abbildungen

nicht den gesamten Lösungsraum, sondern nur die während der Untersuchungen in Kapitel 4.5 simulierten Lösungen dar. Dies sind im Einzelnen:

- Abbildung B.3 für $\mu = 0.014$, basierend auf 8115 simulierten Lösungen.
- Abbildung B.4 für $\mu = 0.042$, basierend auf 8728 simulierten Lösungen.
- Abbildung B.5 für $\mu = 0.056$, basierend auf 4554 simulierten Lösungen.
- Abbildung B.6 für $\mu = 0.07$, basierend auf 1761 simulierten Lösungen.

Für die näherungsweise Darstellung des Lösungsraums für Reibungskoeffizienten größer $\mu = 0.07$ sind nicht ausreichend Simulationsergebnisse vorhanden. Bei einem Reibungskoeffizienten von $\mu = 0.056$ oder höher tritt bei einigen Prozesssimulationen durch zu hohe Niederhaltekräfte Rissbildung auf. Dies spiegelt sich bei den Belohnungstermen darin wider, dass $\mathfrak{r}_{\text{wand}}$ sehr negative Werte annimmt. Diese Lösungen liegen außerhalb des dargestellten Bereichs, der für die entsprechende Achse auf den Minimalwert $-1$ begrenzt ist.

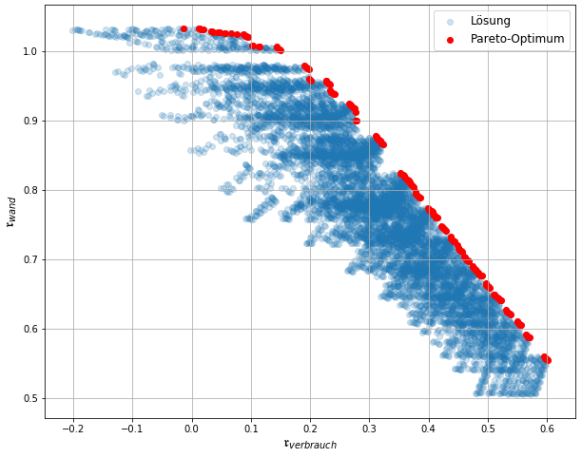



Abbildung B.3: Erzielte Belohnungsterme $\mathfrak{r}_{\text{wand}}, \mathfrak{r}_{\text{verbrauch}}$ pro Lösung für den Reibungskoeffizienten $\mu = 0.014$.

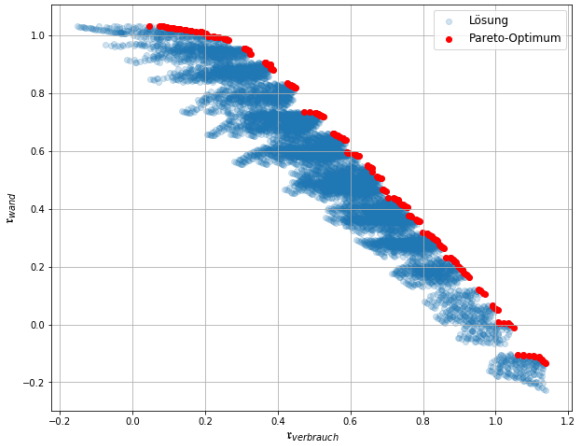



Abbildung B.4: Erzielte Belohnungsterme $r_{\text{wand}}, r_{\text{verbrauch}}$ pro Lösung für den Reibungskoeffizienten $\mu = 0.042$.

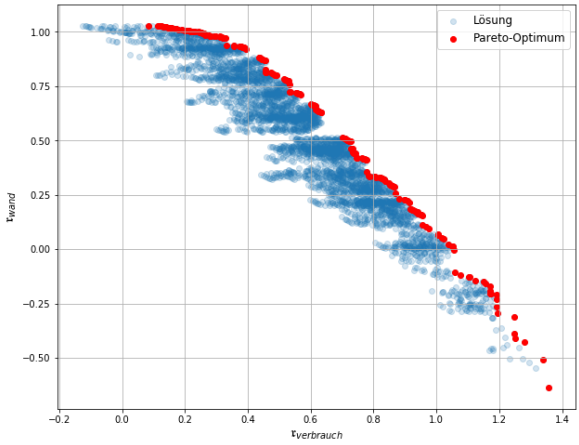



Abbildung B.5: Erzielte Belohnungsterme $r_{\text{wand}}, r_{\text{verbrauch}}$ pro Lösung für den Reibungskoeffizienten $\mu = 0.056$.

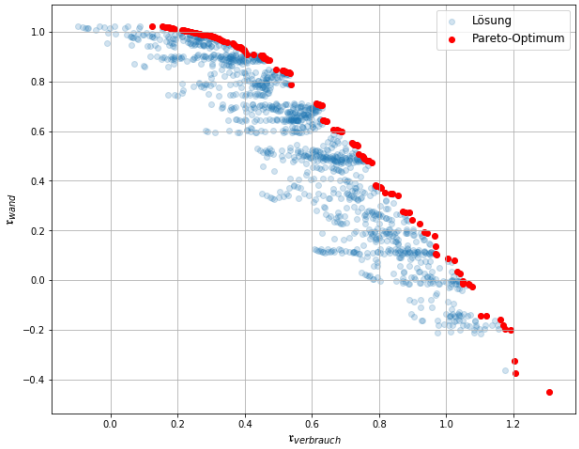



Abbildung B.6: Erzielte Belohnungsterme $r_{\text{wand}}, r_{\text{verbrauch}}$ pro Lösung für den Reibungskoeffizienten $\mu = 0.07$.