STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG

BERND WASCHNECK

Autonome Entscheidungsfindung in der Produktionssteuerung komplexer Werkstattfertigungen

Universität Stuttgart

Fraunhofer IPA

STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG BAND 108

Herausgeber:
Univ.-Prof. Dr.-Ing. Thomas Bauernhansl
Univ.-Prof. Dr.-Ing. Kai Peter Birke
Univ.-Prof. Dr.-Ing. Marco Huber
Univ.-Prof. Dr.-Ing. Dipl.-Kfm. Alexander Sauer
Univ.-Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl
Univ.-Prof. a. D. Dr.-Ing. Prof. E.h. Dr.-Ing. E.h. Dr. h.c. mult. Engelbert Westkämper

Bernd Waschneck

Autonome Entscheidungsfindung in der Produktionssteuerung komplexer Werkstattfertigungen

FRAUNHOFER VERLAG

Kontaktadresse:
Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon 07 11/9 70-11 01
info@ipa.fraunhofer.de; www.ipa.fraunhofer.de

STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG

Herausgeber:
Univ.-Prof. Dr.-Ing. Thomas Bauernhansl[1,2]
Univ.-Prof. Dr.-Ing. Kai Peter Birke[1,4]
Univ.-Prof. Dr.-Ing. Marco Huber[1,2]
Univ.-Prof. Dr.-Ing. Oliver Riedel[3]
Univ.-Prof. Dr.-Ing. Dipl.-Kfm. Alexander Sauer[1,5]
Univ.-Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl[3]
Univ.-Prof. a. D. Dr.-Ing. Prof. E.h. Dr.-Ing. E.h. Dr. h.c. mult. Engelbert Westkämper[1,2]

[1] Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart
[2] Institut für Industrielle Fertigung und Fabrikbetrieb (IFF) der Universität Stuttgart
[3] Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW) der Universität Stuttgart
[4] Institut für Photovoltaik (IPV) der Universität Stuttgart
[5] Institut für Energieeffizienz in der Produktion (EEP) der Universität Stuttgart

Titelbild: © Infineon Technologies AG

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über www.dnb.de abrufbar.
ISSN: 2195-2892
ISBN (Print): 978-3-8396-1623-9

D 93

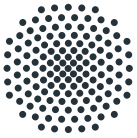
Zugl.: Stuttgart, Univ., Diss., 2020

Druck: Mediendienstleistungen des Fraunhofer-Informationszentrum Raum und Bau IRB, Stuttgart
Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.

© by FRAUNHOFER VERLAG, 2020
Fraunhofer-Informationszentrum Raum und Bau IRB
Postfach 80 04 69, 70504 Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon 07 11 9 70-25 00
Telefax 07 11 9 70-25 08
E-Mail verlag@fraunhofer.de
URL http://verlag.fraunhofer.de

Alle Rechte vorbehalten

Dieses Werk ist einschließlich aller seiner Teile urheberrechtlich geschützt. Jede Verwertung, die über die engen Grenzen des Urheberrechtsgesetzes hinausgeht, ist ohne schriftliche Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Speicherung in elektronischen Systemen.
Die Wiedergabe von Warenbezeichnungen und Handelsnamen in diesem Buch berechtigt nicht zu der Annahme, dass solche Bezeichnungen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und deshalb von jedermann benutzt werden dürften. Soweit in diesem Werk direkt oder indirekt auf Gesetze, Vorschriften oder Richtlinien (z.B. DIN, VDI) Bezug genommen oder aus ihnen zitiert worden ist, kann der Verlag keine Gewähr für Richtigkeit, Vollständigkeit oder Aktualität übernehmen.



Universität Stuttgart

Autonome Entscheidungsfindung in der Produktionssteuerung komplexer Werkstattfertigungen

Von der Graduate School of Excellence advanced Manufacturing Engineering der Universität Stuttgart zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von

Bernd Waschneck

aus Heidelberg

Hauptberichter: Prof. Dr.-Ing. Thomas Bauernhansl

Mitberichter: Prof. Dr. rer. nat. Stefan Wrobel

Tag der mündlichen Prüfung: 27.2.2020

Institut für Industrielle Fertigung und Fabrikbetrieb
der Universität Stuttgart

2020

Vorwort des Autors

Eine Promotion ist primär eine Forschungsarbeit, aber für den Doktoranden wird sie zu einem prägenden Teil des Lebens. Ich habe die Freiheit genossen, in einem offenen, kreativen Umfeld am eigenen Forschungsprojekt zu arbeiten. Mein Dank gilt allen, die dieses Umfeld ermöglicht haben, mir geholfen haben und von denen ich in dieser Zeit etwas lernen durfte.

Besonders bedanken möchte ich mich bei Herrn Univ.-Prof. Dr.-Ing. Thomas Bauernhansl, Leiter des Fraunhofer-Instituts für Produktionstechnik und Automatisierung IPA in Stuttgart sowie des Instituts für Industrielle Fertigung und Fabrikbetrieb (IFF) der Universität Stuttgart, für seine Unterstützung, sein Vertrauen und seine wissenschaftliche Führung während des Forschungsprojekts.

Beiden Mitberichtern der Dissertation möchte ich für ihre sorgfältige Durchsicht und intensive Auseinandersetzung mit der Arbeit danken. Herr Prof. Dr.-Ing. habil. Bernhard Mitschang, Leiter der Abteilung Anwendersoftware des Instituts für Parallele und Verteilte Systeme IPVS und Vorstandsvorsitzender der Graduate School advanced Manufacturing Engineering (GSaME), unterstützte das Forschungsprojekt während der vollen Laufzeit im GSaME Thesis Committee. Herrn Prof. Dr. Stefan Wrobel, Leiter des

Fraunhofer-Instituts für Intelligente Analyse- und Informationssysteme IAIS und Professor für Informatik an der Universität Bonn, möchte ich für die Übernahme des Mitberichts und für sein Interesse an der Arbeit danken.

Die GSaME war entscheidend für die Durchführung dieser Arbeit und für meine persönliche Weiterbildung. Herausheben möchte ich Frau Prof. Dr.-Ing. Sylvia Rohr, Geschäftsführerin der GSaME, die durch Ihren großen Einsatz die GSaME kontinuierlich weiterentwickelt und das hohe wissenschaftliche Niveau in Ausbildung und Forschung sicherstellt. Bei Herrn Dipl.-Ing. Hans-Friedrich Jacobi, Clusterkoordinator des GSaME-Clusters A *"Strategien der wettbewerbsfähigen und nachhaltigen Produktion"*, bedanke ich mich für die Organisation des Cluster-Kolloquiums, spannende Diskussionen und hilfreiche Anmerkungen zu meiner Arbeit.

Das Unternehmen Infineon Technologies AG hat das Forschungsprojekt initiiert, begleitet und in allen Belangen unterstützt. Mein besonderer Dank gilt meinem Betreuer Dr. Thomas Altenmüller, der immer vom Erfolg des Projekts überzeugt war und von dem ich in der Zeit unglaublich viel gelernt habe. Für sein tiefes Vertrauen und seine Unterstützung möchte ich mich besonders bedanken.

Dr. Andreas Kyek möchte ich für seinen fachlichen und persönlichen Rat danken. Ohne die gemeinsamen Kaffeepausen, meist verbunden mit einer anregenden, wissenschaftlichen Diskussion, gäbe es diese Arbeit nicht.

I want to thank Conceição Caldeira for believing in me and giving me the opportunity to work on various topics. I learnt a lot about soft skills and interpersonal communication from you.

Den Kollegen bei Infineon danke ich für die erfolgreiche Zusammenarbeit und die schöne, gemeinsame Zeit. Das Unternehmen wird auch nach Abschluss der Promotion meine berufliche Heimat bleiben und ich freue mich auf die kommenden Jahre in diesem dynamischen, innovativen Umfeld.

Meinen Eltern Barbara und Christian Waschneck und meinen Geschwistern Katja und Anke bin ich dankbar für den vorbehaltlosen Rückhalt und die

bedingungslose Unterstützung. Meiner Frau Katja danke ich für ihre Liebe und dafür, dass sie immer für mich da ist.

Ihnen allen widme ich diese Arbeit.

Neubiberg, August 2018 Bernd Waschneck

Kurzfassung

Die Variabilität in der kundenindividuellen Massenproduktion stellt eine enorme Herausforderung für die industrielle Fertigung dar. Die komplexe Werkstattfertigung als Produktionsprinzip eignet sich aufgrund der inhärenten Flexibilität besonders für die kundenindividuelle Massenproduktion. Allerdings sind die bestehenden Methodiken für die Produktionssteuerung einer Werkstattfertigung für die Einmal- oder Wiederholproduktion ausgelegt, was zu Defiziten in der Massenproduktion führt. Entweder ist die globale Qualität der Ergebnisse suboptimal oder die notwendige Echtzeitfähigkeit in der Entscheidungsfindung kann nicht bereitgestellt werden. Zudem entsteht durch Veränderungen und Anpassungen der Produktionssteuerung einer komplexen Werkstattfertigung ein hoher manueller Aufwand.

In der vorliegenden Arbeit wird eine Methodik für eine dezentrale, selbstorganisierte und autonome Produktionssteuerung für eine Werkstattfertigung entwickelt, die dazu beiträgt, mit der zunehmenden Komplexität und dem Produktionsvolumen umzugehen.

Dabei wird die Produktion als *Reinforcement-Learning*-Modell formalisiert, das die Grundlage für das autonome Lernen einer Strategie zur Optimierung der Abarbeitungsreihenfolge bildet. Mehrere kooperative *Deep-Q-Network*-Agenten werden in diesem Modell darauf trainiert, eine Strategie zu finden, die eine gegebene Bewertungsfunktion – meist ein *Key Performance Indicator*

aus der Produktion – maximiert. Die Neuronalen Netze, in denen die erlernte Entscheidungslogik der *Deep-Q-Network*-Agenten abgebildet ist, werden nach der Trainingsphase in die Produktion übertragen. Der Multi-Agenten-Ansatz trägt dazu bei, dass der Lernvorgang beschleunigt wird und im produktiven Einsatz durch die Dezentralität Entscheidungen schneller bestimmt werden können.

Die Erprobung der Methodik in zwei praxisnahen Fallbeispielen aus der Halbleiterindustrie zeigt ihre Leistungsfähigkeit. In beiden Fallbeispielen konnten Strategien zur Optimierung der Abarbeitungsreihenfolge auf oder über Expertenniveau autonom erlernt werden. Konkret konnte dadurch im zweiten Fallbeispiel der Anteil verspäteter Aufträge in einer Technologieklasse von 17,0 % auf 1,3 % reduziert werden.

Abgerundet wird die Arbeit durch eine Einordnung in das soziotechnische System „Fabrik“, in der die Umsetzung der Reihenfolgeentscheidungen durch die Werker betrachtet wird. Dabei wird offensichtlich, dass die Optimierung der Produktionssteuerung ganzheitlich unter Einbeziehung der Werker in einem kontinuierlichen Verbesserungsprozess erfolgen muss.

Abstract

The variability in mass customization poses an enormous challenge for industrial production. Due to the inherent flexibility the manufacturing principle of a complex job shop is especially well suited for mass customization. However, the methodologies for production control of a job shop are designed for one-off and batch production, which leads to deficits in mass production. Either the global quality of results is suboptimal or the necessary real-time capability is not ensured. Additionally, changes in the production control lead to a high manual effort in the implementation.

In this thesis, a methodology for a decentralized, self-organized and autonomous production control for job shops is developed, which contributes to the management of the rising complexity and production volume.

Production is formalized as a reinforcement learning model which is the foundation for the autonomous learning of a strategy for production dispatching. In this model, several cooperative *Deep Q Network* agents are trained to find a new strategy for maximizing a reward function – mostly a key performance indicator of the production. The neural networks of the *Deep Q Network* agents, in which the scheduling decision logic is mapped, are transferred into production after the training phase. The multi-agent approach contributes to the acceleration of the training and to the real-time capability in productive use due to the decentral architecture.

The validation in two practically relevant use cases proved the performance capability of the methodology. In both use cases new strategies for optimization of the production sequence could be learnt autonomously on par or above the benchmark by human experts. Specifically, the ratio of delayed orders in one technology class was reduced from 17.0 % to 1.3 % in the second example case.

The thesis is concluded by considering dispatching in the context of the sociotechnical system “factory”. In order to improve production control holistically and sustainably, operators have to be involved in a continuous improvement process.

Inhaltsverzeichnis

Abkürzungsverzeichnis **xv**

Abbildungsverzeichnis **xxi**

Tabellenverzeichnis **xxv**

1. Einleitung **1**
- 1.1. Ausgangssituation 1
- 1.2. Problemstellung 6
- 1.3. Zielsetzung und Forschungsfrage 8
- 1.4. Wissenschaftstheoretische Positionierung 10
  - 1.4.1. Einordnung in die Wissenschaft 10
  - 1.4.2. Entdeckungszusammenhang 12
  - 1.4.3. Begründungszusammenhang 14
  - 1.4.4. Verwendungszusammenhang 15
- 1.5. Forschungsprozess und Aufbau der Arbeit 16

2. Grundlagen der Produktion: komplexe Werkstattfertigung **19**
- 2.1. Definition und Abgrenzung von automatischen und autonomen Systemen 20

2.2. Fertigungsprinzipien: Einordnung der komplexen Werkstattfertigung 20
2.3. Eigenfertigungsplanung und -steuerung 26
2.4. Logistische Kerngrößen, Zusammenhänge und Betriebskennlinie in der Werkstattfertigung 32

3. Relevante Verfahren des Maschinellen Lernens 39
3.1. Neuronale Netze 41
3.2. *Reinforcement Learning* 43
3.2.1. Markow-Entscheidungsprozess 44
3.2.2. Vergleich unterschiedlicher RL-Agenten 46
3.2.3. Der *Deep-Q-Network*-Agent 53
3.3. Charakteristiken von Multi-Agenten-Systemen und -Umgebungen 59

4. Stand der Technik in der Halbleiterfertigung 63
4.1. Halbleiterfertigungsprozess 64
4.2. Beispiel: Produktionssteuerung in der Halbleiterscheibenfertigung 69

5. Reflexion bestehender Ansätze zur Produktionssteuerung 75
5.1. Bewertungskriterien einer Produktionssteuerung 76
5.2. Klassische Ansätze der Produktionssteuerung 79
5.2.1. *Dispatching*-Heuristiken 79
5.2.2. Multi-Agenten-Systeme 81
5.2.3. Hyperheuristiken und Maschinelles Lernen 82
5.2.4. Mathematische Optimierung 85
5.2.5. Hybride Systeme aus Heuristiken und mathematischer Optimierung 87
5.3. Zwischenfazit und Defizitbewertung 88

6. Lösungsansatz: Methodik zur autonomen Produktionssteuerung 91
6.1. Simulationsmodell als Grundlage der Methodik 92

6.2. Definition des Zustands- und Aktionsraums 94
6.3. Überwachtes und bestärkendes Lernen in der Fabrik -simulation 99
6.3.1. Überwachtes Lernen in der Fabriksimulation 100
6.3.2. Bestärkendes Lernen in der Fabriksimulation 102
6.4. Multi-Agenten-System in der Produktionssteuerung 106
6.5. Einsatz des Systems in der Produktion 109

7. Prototypische Implementierung der Methodik in einer Halbleiterscheibenfertigung 113
7.1. Aufbau und Parametrisierung des Simulationsmodells . . . 114
7.2. Qualität des Simulationsmodells 118
7.3. Implementierung des Maschinellen Lernens 121
7.4. Beschleunigung des Maschinellen Lernens 122
7.5. Integration in die bestehende Produktionssteuerung einer Halbleiterfertigung 123
7.6. Interpretierbarkeit der Ergebnisfindung 125

8. Fallbeispiele und kritische Bewertung 129
8.1. Fallbeispiel zur Auslastungsoptimierung 130
8.1.1. Simulierte Produktionslinie 130
8.1.2. Benchmark-Heuristiken im Simulationsmodell 134
8.1.3. Wahl der Hyperparameter 134
8.1.4. Formulierung der Bewertungsfunktion 135
8.1.5. Resultate 138
8.2. Fallbeispiel zur Liefertreue 144
8.2.1. Produktionslinie und Hyperparameter 144
8.2.2. Formulierung der Bewertungsfunktion 147
8.2.3. Resultate 148
8.3. Untersuchung des Hyperparameterraums 152
8.4. Kritische Bewertung 155
8.4.1. Reflexion der Methodik 155
8.4.2. Reflexion bezüglich der Intelligenz des Systems . . . 156

9. Soziotechnische Einordnung **159**
9.1. Compliance in Halbleiterfabriken in der Literatur 161
9.2. Relevante Fabrikinformationen, *Dispatching*-Regeln und -Compliance 162
9.3. Ergebnisse der Feldstudie zur *Dispatching*-Compliance . . . 167
9.4. Diskussion der Ergebnisse der Feldstudie 174

10. Zusammenfassung und Ausblick **177**
10.1. Zusammenfassung 177
10.2. Ausblick 179

A. Informationen zum Autor **183**
A.1. Tabellarischer Lebenslauf 183
A.2. Publikationen des Autors 185

Literaturverzeichnis **189**

Abkürzungsverzeichnis

AAAI	*Association for the Advancement of Artificial Intelligence*
ACO	*Ant Colony Optimization*
AMAT	Applied MATerials, Inc.
AMHS	*Automated Material Handling System*
ASIC	*Application-Specific Integrated Circuit*
B2B	*Business to Business*
B2C	*Business to Customer*
BE	*Backend*
CAGR	*Compound Annual Growth Rate*
CAPEX	*Capital Expenditure*
CIM	*Computer Integrated Manufacturing*
CoS	*Cost of Sales*
CP	*Constrained Programming*
CPU	*Central Processing Unit*
D	Deutschland
DAX	Deutscher Aktienindex

DB	Datenbank
DEP	*Deposition* (dt. Abscheidung)
DL	*Deep Learning*
DP	*Dynamic Programming*
DQfD	*Deep Q-learning from Demonstrations*
DQN-Agent	*Deep-Q-Network*-Agent
DRL	*Deep Reinforcement Learning*
DZ	Durchlaufzeit
EPA	Einzelprozessanweisung
ESIA	*European Semiconductor Industry Association*
F&E	Forschung & Entwicklung
FE	*Frontend*
FF	Flussfaktor
FhG	Fraunhofer-Gesellschaft
FIFO	*First-In-First-Out*
FOUP	*Front Opening Unified Pod*
Fraunhofer IPA	Fraunhofer-Institut für Produktionstechnik und Automatisierung
FUR	*Furnace* (dt. Ofentechnik)
GPU	*Graphics Processing Unit*
GSaME	*Graduate School of Excellence advanced Manufacturing Engineering* der Universität Stuttgart
IBM	International Business Machines Corporation
IC	*Integrated Circuits*
IFAG	Infineon Technologies AG
IFAT	Infineon Technologies Austria AG
IFRS	*International Financial Reporting Standards*

IFX	Tickersymbol der Infineon Technologies AG an der Frankfurter Börse
ILOG	*Intelligence Logiciel, part of IBM*
IMP	*Implantation* (dt. Ionenimplantation)
IoT	*Internet of Things*
IT	Informationstechnik
KI	Künstliche Intelligenz
KPI	*Key Performance Indicator*
KR	Kritischer Rationalismus
LIME	*Local Interpretable Model-agnostic Explanation*
LIT	*Lithography* (dt. Fotolithografie)
LP	Linear Programming
LR	*Learning Rate*
MAS	Multi-Agenten-System
max	maximiere
MDT	Mechanismus-Design-Theorie
MEMS	*MicroElectroMechanical System*
MES	*Manufacturing Execution System*
MET	*Metrology* (dt. Messtechnik)
MILP	*Mixed Integer Linear Programming*
min	minimiere
ML	Maschinelles Lernen
MSE	*Mean Squared Error* (dt. mittlere quadratische Abweichung)
NN	Neuronale Netze
NP	Nichtdeterministische Polynomialzeit
OCM	*Operating Curve Management*

ODD	*Operation Due Date*
OEE	*Overall Equipment Efficiency*
OEM	*Original Equipment Manufacturer*
OHT	*Overhead Hoist Transfer*
OR	*Operations Research*
PET	*Product Engineering Tools* (dt. Anlagen der Produktentwicklung)
ReLU	*Rectified Linear Unit*
RFID	*Radio-Frequency IDentification*
RL	*Reinforcement Learning*
RPZ	Rohprozesszeit
RTD	*Real-Time Dispatcher*
SC	*Supply Chain*
SECS/GEM	*SEMI Equipment Communications Standard/ Generic Equipment Model*
SEMI	*Semiconductor Equipment and Materials International*, Wirtschaftsverband
SGD	*Stochastic Gradient Descent*
SIA	*Semiconductor Industry Association*
SiP	*System-in-Package*
SL	*Supervised Learning*
SoC	*System-on-Chip*
SQL	*Structured Query Language*
TK	Technologieklasse
TNU	*Target Network Update*
TPU	*Tensor Processing Unit*
TSMC	Taiwan Semiconductor Manufacturing Company, Limited

t-SNE	*t-distributed Stochastic Neighbor Embedding*
UL	*Unsupervised Learning*
USP	*Unique Selling Proposition*
WET	*Wet Chemistry* (dt. Nasschemie)
WHD	*Wafer Handling* (dt. Scheibenhandhabung)
WSC	*World Semiconductor Council*

Abbildungsverzeichnis

1.1. Entwicklung der Produktion bezüglich Produktionsvolumen und -vielfalt 2
1.2. Unterschiedliche Entwicklungsrichtungen in der Halbleiterindustrie bezüglich Technologie und Produkte 5
1.3. Größe und Wachstumsraten verschiedener Segmente im Halbleitermarkt 6
1.4. Weltweite Investausgaben in der Halbleiterindustrie 7
1.5. Strukturierung von Wissenschaft 11
1.6. Forschungsmethodisches Vorgehen 14
1.7. Forschungsphasen nach Ulrich und Struktur der Arbeit . . 17

2.1. Wellen im Umlaufbestand in der Halbleiterfertigung 24
2.2. Schematische Darstellung der Merkmale einer komplexen Werkstattfertigung 25
2.3. Eigenfertigungsplanung und -steuerung 27
2.4. Bezeichnung und Einteilung der Maschinenzustände 34
2.5. Betriebskennlinie für verschiedene α-Werte 37

3.1. Überblick über die vorgestellten Bereiche der KI 41
3.2. Beispiel eines NNs mit zwei verdeckten Ebenen 43

3.3. Umgebung und Agent in einem MARKOW-Entscheidungsprozess 45
3.4. Funktionsweise des DQN-Agenten 54
3.5. Charakterisierung von MAS und der Umgebung 60

4.1. Größenordnungen vom Wafer zum einzelnen Transistor . . . 65
4.2. Erster Teil der Lieferkette der Halbleiterfertigung (*Frontend*) 66
4.3. Zweiter Teil der Lieferkette der Halbleiterfertigung (*Backend*) 66
4.4. Strukturierungsprozess des *Gate*-Oxids 68
4.5. Gegenüberstellung von *Bay-Chase*- und *Ballroom*-Layout . . . 70
4.6. Blick in einen *Intrabay* in der Reinraumproduktion 71
4.7. Automatisierung in der Halbleiterfertigung 72
4.8. Produktionssteuerung in der INFINEON Waferprozessierung Dresden 74

5.1. Bewertung von Ansätzen zur Produktionssteuerung einer komplexen Werkstattfertigung 88

6.1. MARKOW-Entscheidungsprozess in der Fabriksimulation . . . 94
6.2. Zusammenspiel zwischen Fabriksimulation und Eventverarbeitung durch einen MARKOW-Entscheidungsprozess 101
6.3. Entscheidungsfindung und Training der *Policy* π in der Produktionssteuerung 103
6.4. Vereinfachter Ablauf des Lernprozesses in der Fabriksimulation 105
6.5. Anlernen der DQN-Agenten im MAS und Übertragung der NNs in die Produktion 109

7.1. Datenfluss zwischen den Hauptklassen der Fabriksimulation 115
7.2. Klassen des *P*-Pakets der Fabriksimulation 117
7.3. Synchronisierung des Digitalen Zwillings über ein wissensbasiertes Änderungsmanagementsystem 120

7.4. Bildschirmfoto des Applied Materials, Inc. APF *Formatters* 124

8.1. Übersicht über das Simulationsmodell für Fallbeispiel 1 mit drei Technologieklassen und vier Maschinengruppen 132
8.2. Charakterisierung der drei Szenarien in der Fabriksimulation 133
8.3. Verlustfunktion der NNs in Phase A für die drei untersuchten Szenarien in Fallbeispiel 1 für alle drei Agenten 139
8.4. Bewertungsfunktion in Phase A in den drei untersuchten Szenarien in Fallbeispiel 1 140
8.5. Mittlerer *Q*-Wert während des Trainings in Phase A in den drei untersuchten Szenarien in Fallbeispiel 1 141
8.6. Bewertungsfunktion in Phase B in den drei untersuchten Szenarien in Fallbeispiel 1 142
8.7. Mittlerer *Q*-Wert während des Trainings in Phase B in den drei untersuchten Szenarien in Fallbeispiel 1 143
8.8. Übersicht über das Simulationsmodell für Fallbeispiel 2 mit drei Technologieklassen und vier Maschinengruppen 146
8.9. Verlauf der *ϵ-greedy policy* in Phase A in Fallbeispiel 2 147
8.10. Verlustfunktion der NNs in Phase A in Fallbeispiel 2 149
8.11. Bewertungsfunktion in Phase A in Fallbeispiel 2 149
8.12. Mittlerer *Q*-Wert während des Trainings in Phase A in Fallbeispiel 2 150
8.13. Histogramme der DZ der produzierten Aufträge mit den entsprechenden *Dispatching*-Mechanismen 151
8.14. Verlustfunktion des NNs für unterschiedliche *learning-rate*- und *target-network-update-frequency*-Parameter 153
8.15. *Q*-Wert der DQN-Agenten für unterschiedliche *learning-rate*- und *target-network-update-frequency*-Parameter 153
8.16. Bewertung in der RL-Umgebung für unterschiedliche *learning-rate*- und *target-network-update-frequency*-Parameter 154

9.1. Histogramm der Warteschlangenlänge der Standorte 168
9.2. Durchschnittliche absolute Compliance C_{abs} in Abhängigkeit von der Warteschlangenlänge 169
9.3. Durchschnittliche relative Compliance C_{rel} in Abhängigkeit von der Warteschlangenlänge 169
9.4. Vergleich der mittleren relativen Compliance der Fabriken in Abhängigkeit der wichtigsten *Dispatching*-Regeln 171
9.5. Boxplot der Warteschlangenlänge an unterschiedlichen Maschinengruppen. 171
9.6. Boxplot der relativen Compliance C_{rel} an unterschiedlichen Maschinengruppen 173
9.7. Autokorrelation der relativen *Dispatching*-Compliance und Korrelationskoeffizienten an Maschinengruppen 173

10.1. Einordnung der autonomen Produktionssteuerung im S-Kurven-Konzept 181

Tabellenverzeichnis

1.1. Zielsetzung der Arbeit 9

2.1. Grundsätzliche Eignung unterschiedlicher Fertigungsprinzipien für die Produktionsarten 21
2.2. Zielgrößen der Logistikleistung und der Logistikkosten 27

3.1. Überblick über *Reinforcement-Learning*-Agenten 52
3.2. Erster Teil der Hyperparameter und der verwendeten Werte in der ersten Publikation des DQN-Agenten 57
3.3. Zweiter Teil der Hyperparameter und der verwendeten Werte in der ersten Publikation des DQN-Agenten 58

8.1. Rüstzeiten für das Umrüsten von einer Technologieklasse (TK) zur anderen am *Implanter* 131
8.2. Vergleich der Hyperparameter des DQN-Agenten-Algorithmus, Teil 1 136
8.3. Vergleich der Hyperparameter des DQN-Agenten-Algorithmus, Teil 2 137
8.4. Vergleich der DQN-Agenten-Steuerung mit den *Dispatching*-Heuristiken in der Testumgebung 144

8.5. Rüstzeiten für das Umrüsten von einer Technologieklasse (TK) zur nächsten am *Implanter* 145
8.6. Vergleich der Liefertreue in Bezug auf eine Steuerung durch *Dispatching*-Heuristiken und den DQN-Agenten-Ansatz 150

9.1. Abkürzungsverzeichnis und Übersetzungen der Bezeichnungen der Maschinengruppen in der Halbleiterindustrie 172

Kapitel 1

Einleitung

1.1. Ausgangssituation

Kundenindividuelle Massenfertigung und Personalisierung von Produkten sind wesentliche Strömungen der industriellen Produktion im 21. Jahrhundert, die starke Auswirkungen auf das Fertigungsprinzip haben (Bauernhansl et al., 2014, S. 13; Jovane et al., 2003; siehe auch Abbildung 1.1). Die Werkstattfertigung, meist mit der Manufaktur und Handfertigung assoziiert, wird als hochautomatisierte und digitalisierte Variante für die personalisierten Massenproduktion eingesetzt (Kern et al., 2015; Waschneck et al., 2016a), denn sie bietet die notwendige Flexibilität für die Personalisierung. Durch die Massenproduktion in der Werkstatt ändert sich der Charakter der Werkstattfertigung fundamental. Sie wird in dieser modernen Variante als komplexe Werkstattfertigung bezeichnet (Mason et al., 2002; Mönch et al., 2013, S. 25). Die dadurch verursachten Veränderungen haben insbesondere Auswirkungen auf die Produktionssteuerung. In dieser Arbeit wird eine Methodik entwickelt, wie eine autonome Produktionssteuerung eine derartige Werkstattfertigung lenkt.

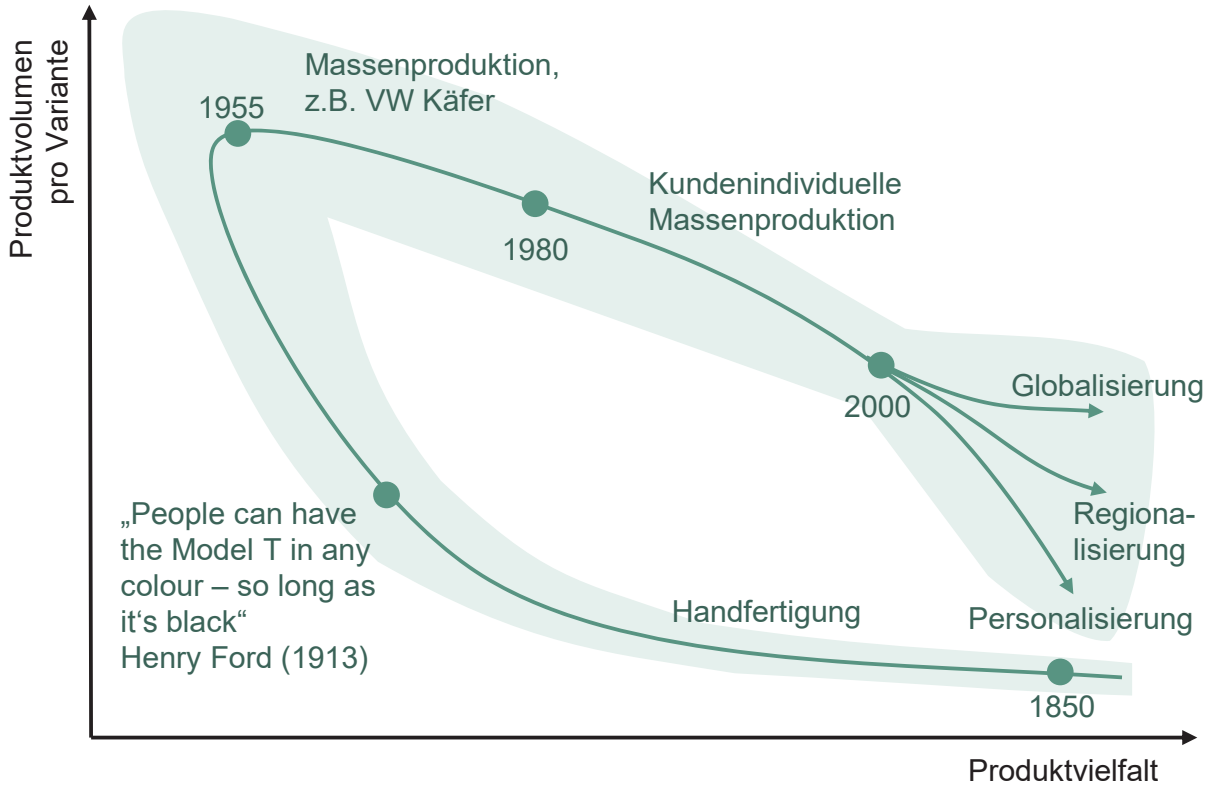


Abbildung 1.1.: Entwicklung der Produktion bezüglich Produktionsvolumen und -vielfalt. Quelle: Bauernhansl et al. (2014, S. 13) und Jovane et al. (2003)

Die Methodik ist insbesondere für die Halbleiterindustrie relevant, die traditionell den Produktionstyp der komplexen Werkstattfertigung nutzt. Einerseits stoßen die traditionellen Maßnahmen zur Produktivitätssteigerung in der Halbleiterindustrie an ihre Grenzen (Mönch et al., 2011), andererseits ändern sich die Anforderungen des Marktes, der immer mehr spezialisierte Chips, z.B. *Internet of Things (IoT)* Sensoren, erwartet (*International Technology Roadmap for Semiconductors* 2011, S. 10).

Bei den traditionellen Ansätzen zur Produktivitätssteigerungen der Halbleiterindustrie handelt es sich um die Miniaturisierung, die Verbesserung der Ausbeute und um die Vergrößerung der Silizium-Wafer, auf denen die Chips produziert werden (Mönch et al., 2011). Die Miniaturisierung wird durch das Moore'sche Gesetz beschrieben, das Gordon Moore, Mitgründer des Halbleiterkonzerns Intel Corporation, 1965 formulierte. Das Moore'sche

Gesetz besagt, dass sich die Transistorendichte pro Fläche alle 12—24 Monate verdoppelt (Brock et al., 2006, S. 25ff.; Moore 1965). Allerdings stößt das Moore'sche Gesetz in den nächsten Jahren an seine physikalischen Grenzen: Transistoren bestehen aus wenigen Atomlagen und können nicht weiter verkleinert werden (Khan et al., 2018). Außerdem ging mit der Reduktion der Größe immer eine Reduktion des Preises einher. Seit dem 28 nm Technologieknoten ist der Preis pro Transistor jedoch etwa konstant, da die Entwicklungs- und Produktionskosten enorm steigen (Jones, 2016).

Die Ausbeute in der Halbleiterfertigung liegt heute bereits deutlich über 90 %[1]. Verbesserungen im absoluten Niveau der Ausbeute rechtfertigen den notwendigen Aufwand nicht mehr. Fortschritte kommen heute im Wesentlichen aus der beschleunigten Verbesserung der Ausbeute beim Hochfahren neuer Produktionsanlagen oder neuer Technologien[2].

Der Durchmesser der Wafer wurde über die Jahre kontinuierlich bis 300 mm gesteigert. Etwa alle zehn Jahre[3] wurde die Scheibengröße erhöht, bis 1997 die ersten Pilotlinien auf 300 mm starteten (Integrated Circuit Engineering Corp., 1997). Seitdem gab es keine Vergrößerung des Scheibendurchmessers in der Produktion. Das Global 450 Consortium (G450C), welches die Einführung von 450 mm Wafern vorantreiben sollte, hat seine Arbeit aufgrund verschiedener Hürden eingestellt (Hruska, 2017).

Die jährliche Preisreduktion, die von Kunden aufgrund der jahrzehntelangen Preisentwicklung erwartet wird, muss nun auf anderen Wegen erreicht werden. Daher liegt ein Fokus in der Halbleiterindustrie nun auf *Operational Excellence*.

Eine weitere veränderte Anforderung hängt mit der Diversifikation des Produktportfolios in der Halbleiterindustrie zusammen. Mit dem Hintergrund, dass das Produktportfolio der Halbleiterindustrie lange ausschließlich

[1] Quelle: Internes *Yield-Reporting* der Infineon Technologies AG

[2] Quelle: Interne Analyse der Infineon Technologies AG basierend auf dem *Yield-Reporting* und *Cost of Yield (CoY)*-Daten.

[3] Silizium-Scheiben mit einem Durchmesser von 100 mm wurden 1975 eingeführt. Im Jahr 1980 folgten Wafer mit einem Durchmesser von 150mm. Elf Jahre später, 1991, wurden 200 mm Scheiben verwendet. Ab 1997 starteten erste Pilotlinien mit 300 mm Wafern, die sich dann ab etwa 2001 am Markt etablierten. Quellen: Hattori (2015) und Integrated Circuit Engineering Corp. (1997)

aus Speicher- und Logikchips bestand, wird die enorme Veränderung durch ein breiteres Produktportfolio deutlich.

Es werden zwei Richtungen unterschieden, in die sich die Halbleiterindustrie entwickelt (*International Technology Roadmap for Semiconductors* (2011, S. 10); siehe Abbildung 1.2). Bei *More Moore* handelt es sich um die klassische Miniaturisierung in der Informationsverarbeitung, also Speicher- und Logikchips. Neuer, und entscheidend durch das IoT getrieben, ist die *More-than-Moore*-Richtung: Sensorik und Aktorik. Dieses Gebiet ermöglicht das Verschmelzen von physikalischer und digitaler Welt. Wie in Abbildung 1.3 gezeigt, weisen Marktsegmente mit diesen Produkttypen die größten Wachstumsraten auf. Insbesondere das IoT und der Automobilmarkt sind wichtige Abnehmer für Sensorik und Leistungshalbleiter.

Bei den entsprechenden Produkten, beispielsweise *MicroElectroMechanical Systems (MEMS)*, kann mit einer dem Moore'schen Gesetz folgenden Miniaturisierung keine Kostenreduktion erzielt werden. Gleichzeitig treten sie in einer für Halbleiterhersteller neuen Variantenvielfalt auf. Während bei Speicherbausteinen nur wenige Varianten existieren, werden bei analogen und *mixed-signal* Chips sowie Sensoren und Aktoren häufig applikationsspezifische Produkte[1] gefertigt, die exakt den Kundenanforderungen entsprechen (Barr, 2007, S. 1 ff.; *International Technology Roadmap for Semiconductors* 2011, S. 10).

Der Fokus auf *Operational Excellence* sowie die Veränderungen durch Diversifizierung des Produktionsportfolios führen zu neuen Anforderungen an die Produktionssteuerung in der Halbleiterindustrie. Diese sind nicht spezifisch für das Beispiel Halbleiterindustrie, sondern finden sich in jeder komplexen Werkstattfertigung. Dieser Produktionstyp wird – mit dem Fortschreiten der personalisierten Massenproduktion – weiter an Bedeutung gewinnen.

Innovationen werden entweder durch einen *Market Pull* (Möhrle et al., 2018a), also durch gestiegene Anforderungen des Anwendermarktes, oder durch einen *Technology Push* (Möhrle, 2018) hervorgerufen. Die gestiegenen Anforderungen an die Produktionssteuerung durch den *Market Pull*

[1] In der Halbleiterindustrie werden applikationsspezifische Bausteine als *ASIC (Application-Specific Integrated Circuit)* bezeichnet.

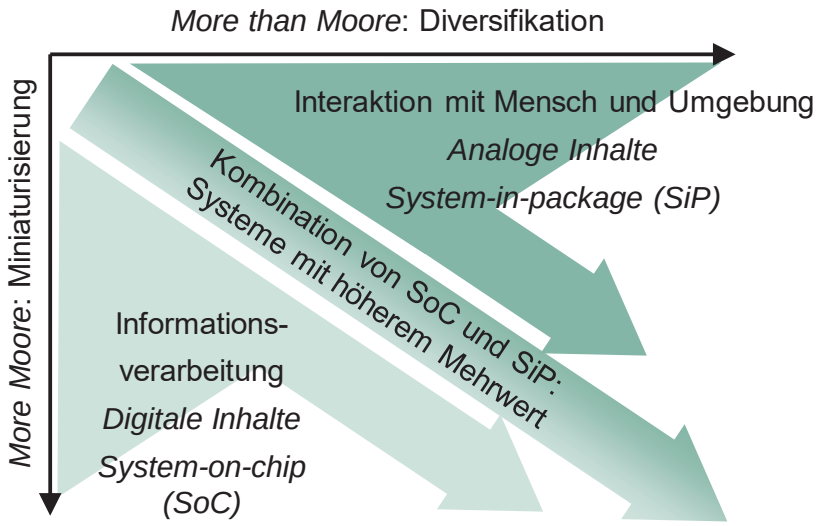


Abbildung 1.2.: Unterschiedliche Entwicklungsrichtungen in der Halbleiterindustrie bezüglich Technologie und Produkte. Quelle: *International Technology Roadmap for Semiconductors* (2011, S. 10)

wurden bereits beschrieben. Im Folgenden wird nun auf die Verfügbarkeit neuer Technologien für die Produktionssteuerung, den *Technology Push*, eingegangen. Durch die Künstliche Intelligenz (KI) und *Deep Learning (DL)* gibt es einen Technologieschub, bei dem mit den Methoden des Maschinellen Lernens (ML) disruptive Technologien zur Verfügung stehen. Nach einer Studie von McKinsey liegt das Wertschöpfungspotential von *Deep-Learning*-Technologien bei bis zu 5,8 Billionen US-Dollar (Chui et al., 2018). Das DL eröffnet technisch neue Möglichkeiten, die in neuen Anwendungen resultieren. So konnten erstmals professionelle Spieler des asiatischen Brettspiels *Go* durch ein Computerprogramm geschlagen werden (Silver et al., 2017b). Die gute Verfügbarkeit von DL durch *Open-Source*-Software-*Frameworks* wie *TensorFlow* (Abadi et al., 2016) erlaubt eine Anwendung in neuen Feldern. In dieser Arbeit wird der Einsatz von ML Techniken in der Produktionssteuerung untersucht.

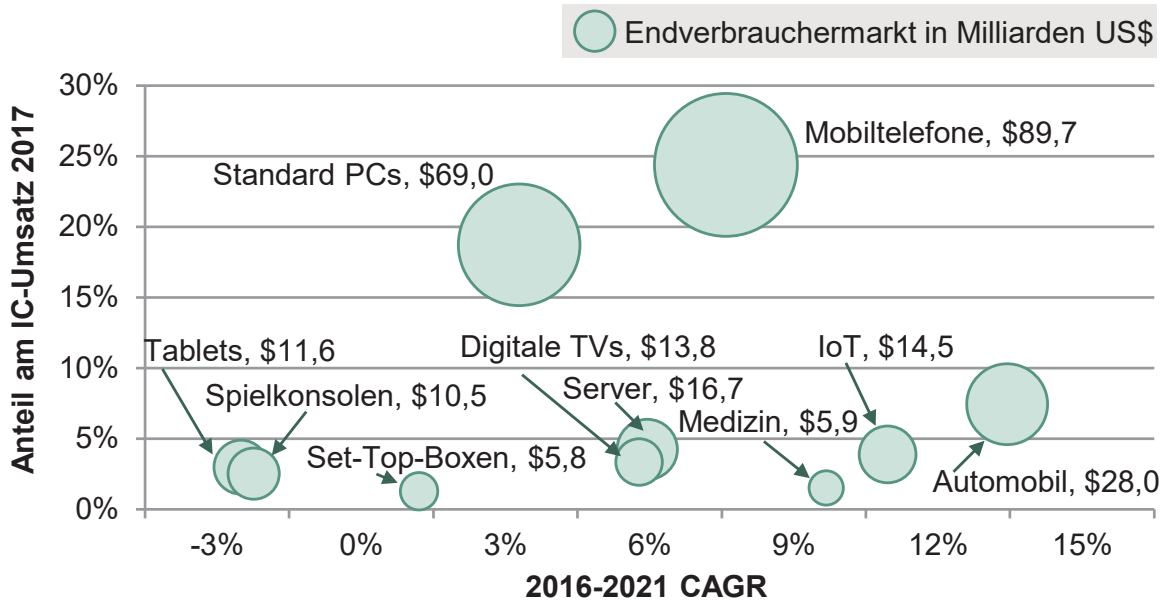


Abbildung 1.3.: Größe und Wachstumsraten verschiedener Segmente im Halbleitermarkt. Basierend auf McClean et al. (2018, Abbildung 5-4)

1.2. Problemstellung

Mit den gestiegenen Anforderungen durch Regionalisierung und Personalisierung von Produkten sowie starker Konkurrenz durch Fertigungsstandorte mit niedrigerem Lohnniveau steigt der Druck auf die Produktion. Produktzyklen werden immer kürzer, weshalb der Geschwindigkeit in der Produktion eine gestiegene Bedeutung zukommt (Chesbrough, 2007). Die Produktionssteuerung ist entscheidend für die Kapazitätsauslastung der Maschinen, die entscheidend für die Kosten ist, sowie für die Logistikleistung, die kurze Durchlaufzeiten (DZs) und eine hohe Termintreue garantieren soll (Ouelhadj et al., 2009; Pinedo, 2016, S. 2). Die effiziente Ausnutzung der vorhandenen Kapazitäten ist die Voraussetzung für eine konkurrenzfähige Fertigung, die termingerecht liefert.

Obwohl die Problemstellung eine allgemeine Bedeutung für komplexe Werkstattfertigungen hat, wurde sie aus der Halbleiterindustrie heraus motiviert. Der Industriezweig ist besonders kapitalintensiv (Ballhaus et al., 2009, S. 10), was an den hohen Investitionsausgaben, gezeigt in Abbildung 1.4,

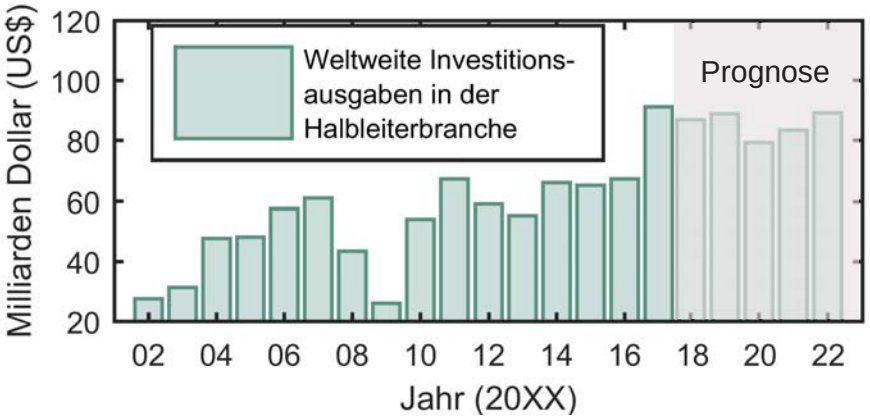


Abbildung 1.4.: Weltweite Investausgaben in der Halbleiterindustrie. Basierend auf McClean et al. (2018, Abbildung 4-1)

deutlich wird. Nach der Prognose von McClean et al. (2018, S. 4-1) werden sich die Investitionsausgaben zwischen 2018 und 2022 auf konstant hohem Niveau bewegen. So beträgt die *Invest-to-Sales* Quote der Infineon Technologies AG beispielsweise 15 %, von denen 13 % auf *Capital Expenditure (CAPEX)-to-Sales* entfallen[1]. Die *Cost-of-Sales (CoS)* Rate nach den *International Financial Reporting Standards (IFRS)* betrug im ersten Quartal 2018[2] 62,9 %. Nachdem in den meisten Absatzmärkten für Halbleiter lange Zeit Innovation und Technologie die Alleinstellungsmerkmale (engl. *Unique Selling Proposition (USP)*) waren, wird der Preis zunehmend ein Unterscheidungsmerkmal (Ballhaus et al., 2009, S. 10).

Neben der Auslastung muss die Produktionssteuerung die termingerechte Fertigstellung des Produkts garantieren. Halbleiterbauelemente stehen meistens am Anfang einer längeren Wertschöpfungskette. So werden beispielsweise in einem modernen Kraftfahrzeug heute mehrere hundert Halbleiterelemente mit einem Gesamtwert von 350$ verbaut (Eisert, 2017). Dieser Wert wird bis 2021 auf 400$ ansteigen (Eisert, 2017). Hunderte unterschiedlicher Halbleiterbauelemente, deren Wert teilweise im €-Cent Bereich liegt, müssen termingerecht an den Automobil-Fließbändern verfüg-

[1] Angekündigt auf dem *Infineon Capital Markets Day* in London am 12. Juni 2018.
[2] Nach Kalenderjahr; nicht Finanzjahr

bar sein. Da die langen DZs in der Halbleiterindustrie stark variieren, wird diese Unsicherheit in der DZ mit größerem Lagerbestand ausgeglichen.

Lange DZs von mehreren Monaten, hohe Kosten für die Produktionskapazitäten, und ein ständig wachsendes Portfolio an Produkten und Produktvarianten führt dazu, dass bestehende Methodiken die Anforderungen zunehmend nicht mehr erfüllen können. In Kapitel 5 wird im Detail auf die veränderten Anforderungen an die Produktionssteuerung sowie auf die bestehenden Methoden eingegangen.

1.3. Zielsetzung und Forschungsfrage

Zielsetzung dieser Arbeit ist die Entwicklung einer Methodik zur Produktionssteuerung komplexer Werkstattfertigungen. Im Einsatz soll sich die Produktionssteuerung durch eine gesteigerte Flexibilität und Transparenz auszeichnen. Kundenindividuelle Massenproduktion in der komplexen Werkstattfertigung soll damit bei gleichbleibenden oder besseren logistischen Zielgrößen im Vergleich zur Massenfertigung einer einzelnen Produktvariante möglich sein.

Die Skalierbarkeit soll durch eine dezentrale Konzeption erreicht werden. Die dezentralen Komponenten verfolgen ein globales Ziel, wodurch eine globale Optimierung der ganzen Linie erreicht wird. Die klare Zieldefinition sorgt für Transparenz in der Fertigungslinie. Die notwendige Flexibilität soll durch selbstlernende, adaptive Komponenten erreicht werden.

Die Forschungsfrage lässt sich aus der beschriebenen Zielsetzung der Arbeit ableiten:

> *„Wie kann die Entwicklung und Umsetzung einer autonomen Steuerungsmethodik dazu beitragen, mit der ständig zunehmenden Komplexität in einer Werkstattfertigung umzugehen?"*

Anwendungs-dimension	– Komplexe Werkstattfertigung – Produktionssteuerung – Dispatching und Scheduling – Entscheidungsfindung
Gestaltungs-dimension	– Erweiterter Betrachtungsrahmen von Produktionssteuerungen (Einflussgrößen, Gestaltung des Zustands- und Aktionsraums) – Einsatz von neuen Methoden in der Optimierung – Konzeption selbstlernender Systeme
Ergebnis-dimension	– Optimale Abarbeitungsreihenfolge in der Produktion – Flexibilität in der Produktionssteuerung – Transparenz in den logistischen Zielgrößen – Autonomes Bestimmen der Steuerungsregeln

Tabelle 1.1.: Zielsetzung der Arbeit in Anlehnung an Bauernhansl (2003, S. 10)

Die Forschungsfrage zielt auch auf die Verbesserung des betriebswirtschaftlichen Erfolgs ab, indem die neue Methodik die Wertschöpfungskette unterstützt. Im Modell nach Porter (1998, S. 36 ff.) wird die Produktionssteuerung als primäre Aktivität im Bereich Operationen/Produktion eingeordnet. Porter unterscheidet in jedem Bereich drei Aktivitätstypen: direkte, indirekte und qualitätssichernde Aktivitäten (Porter, 1998, S. 44). Die Produktionssteuerung stellt eine indirekte Aktivität dar, da sie zur kontinuierlichen Ausführung der Produktion beiträgt[1]. Die Produktionssteuerung garantiert eine Maximierung des Mehrwerts unter Einsatz der vorhandenen, beschränkten Ressourcen. Die Effizienz der Potentialfaktoren, also der Betriebsmittel und der menschlichen Arbeit am Erzeugnis, werden optimiert. Ein Beispiel hierfür ist die Maschinenauslastung: Durch Minimierung der Rüstzeiten auf einer Maschine wird der Durchsatz auf dieser gesteigert. Hierdurch entsteht ein gesteigerter *Output*, also mehr Wertschöpfung, bei gleichem Kapitaleinsatz für die Betriebsmittel.

[1] Porter selbst nennt *Scheduling* als Beispiel für eine indirekte Aktivität, die die kontinuierliche Ausführung eines Primärbereichs garantiert (Porter, 1998, S. 44).

Nach der Darstellung des Stands der Technik werden ausgehend von der Forschungsfrage in Kapitel 5.1 Bewertungskriterien abgeleitet, mit denen der Erfolg der neuen Methodik festgestellt werden kann. Bestehende Verfahren werden in Kapitel 5.3 anhand der Bewertungskriterien beurteilt und in Kapitel 8.4.1 wird schließlich das autonome *Dispatching* diesbezüglich analysiert.

Die Forschungsfrage bestimmt die Gestaltung und den Aufbau des im Folgenden beschriebenen Forschungsprozesses. Die Forschungsfrage bildet damit das Fundament und die Leitlinie für den Erkenntnisgewinn.

1.4. Wissenschaftstheoretische Positionierung

Für einen geordneten Forschungsprozess ist es entscheidend, die Grundlagen, Paradigmen und Positionierungen zu kennen, auf denen der Forschungsprozess aufbaut. In Kapitel 1.4.1 wird das Thema in die Wissenschaftssystematik eingeordnet. Danach folgt die Betrachtung des Entdeckungs-, Begründungs- und Verwendungszusammenhangs, die nach Ulrich et al. (1976a, b) wesentliche Gesichtspunkte der realwissenschaftlichen Forschung bilden.

1.4.1. Einordnung des Forschungsprojekts in die Wissenschaft

Wissenschaften können gemäß ihres Anspruchs und ihrer Zielsetzung strukturiert werden. In dieser Arbeit wird eine Strukturierung nach Kornmeier (2007, S. 14) und Ulrich et al. (1976a) verwendet, die in Abbildung 1.5 gezeigt ist.

Zuerst werden metaphysische von nicht-metaphysischen Wissenschaften unterschieden. Metaphysische Wissenschaften wie die Theologie und Philosophie beschäftigen sich mit den Zusammenhängen des Seins über dem Erfahrbaren.

Die nicht-metaphysischen Wissenschaften lassen sich in Formalwissenschaften und Realwissenschaften aufgliedern. Die Formalwissenschaften beschäftigen sich vorzugsweise mit Methoden und Zeichensystemen. Die Realwissenschaften beschäftigen sich dagegen mit realen, beobachtbaren

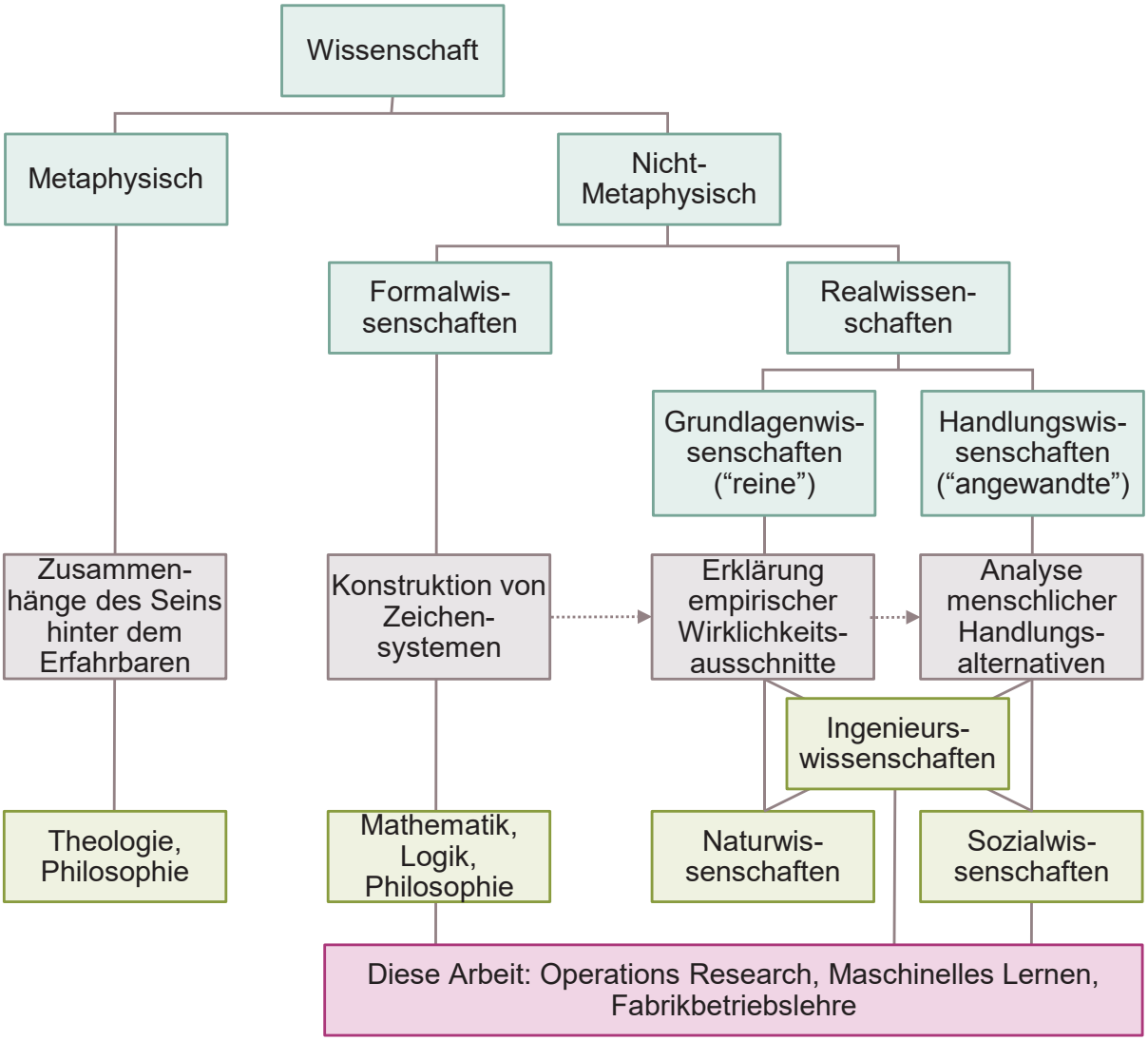


Abbildung 1.5.: Strukturierung von Wissenschaft. Basierend auf Kornmeier (2007, S. 14) und Ulrich et al. (1976a)

Phänomenen. Bei der Analyse und Erklärung dieser Phänomene werden häufig die Methoden und Zeichensysteme der Formalwissenschaft genutzt.

Bei den Realwissenschaften werden Grundlagenwissenschaften und Handlungswissenschaften unterschieden. Während sich die ersteren mit der Erklärung empirischer Wirklichkeitsausschnitte befassen, analysieren Handlungswissenschaften menschliche Handlungsalternativen. Die Ingenieurswissenschaften finden sich zwischen diesen Bereichen.

In dieser Arbeit wird stark auf abstrakte mathematische Methoden der Optimierung und des ML zurückgegriffen. Diese werden im Bereich der Ingenieurswissenschaften angewandt. Zuletzt spielen jedoch auch die Sozialwissenschaften eine Rolle, da Menschen die Entscheidungen des automatisierten Systems umsetzen müssen.

1.4.2. Entdeckungszusammenhang

Mit dem Entdeckungszusammenhang wird der gedankliche Bezugsrahmen des Forschungsprojekts beschrieben (Ulrich et al., 1976a). Damit wird das Vorwissen und Vorverständnis des Forschers erfasst, das im Forschungsprozess die konzeptuelle Basis bildet. Bei der Strukturierung des Entdeckungszusammenhangs folgt das Kapitel Bauernhansl (2003, S. 13), der den Bezugsrahmen in drei Aspekte unterteilt. Den ersten bildet die zugrundeliegende Leittheorie, dann folgt der Erfahrungshorizont des Autors und zuletzt kommen die prägenden Arbeiten im Grundlagenbereich.

Nach Kuhn (1996) baut jeder Wissenschaftsbereich auf Paradigmen auf, die in der Wissenschaftsdisziplin anerkannt sind und meistens nicht hinterfragt werden. Indem die Paradigmen ein Fundament für die Forschung bilden, leiten und beschleunigen sie den wissenschaftlichen Fortschritt. In der Physik, dem Ausbildungshintergrund des Autors, sind Paradigmen und besonders Paradigmenwechsel einfach verständlich: Der Wechsel des geozentrischen Weltbilds zum heliozentrischen Weltbild durch Nikolaus Kopernikus ist ein bedeutendes, sowie einfaches Beispiel für einen Paradigmenwechsel (Kuhn, 1996, S. 68ff.).

Die vorliegende Arbeit folgt dem *systemtheoretischen Ansatz* nach Ulrich (1970). Der praxisnahe, interdisziplinäre Ansatz stellt eine gute Basis für die Ingenieurswissenschaften dar. Insbesondere der Systemansatz des Paradigmas, der psychologische, soziologische, ökonomische und technologische Aspekte integriert, passt zum Charakter des Forschungsprojekts.

Die Arbeit stützt sich auf die Erfahrung des Verfassers innerhalb eines Kooperationsprojekts zwischen der Graduate School of Excellence

advanced Manufacturing Engineering[1] (GSaME) und der Infineon Technologies AG[2]. Einerseits war der Verfasser der Arbeit im Rahmen des Kooperationsprojekts eng in die internen Abläufe, Prozesse und Diskussionen der Infineon Technologies AG eingebunden. Dort war der Autor zunächst in der Zentralabteilung *Manufacturing Excellence*, dann im *Operations Office* und zuletzt in der Abteilung *Frontend Manufacturing Analytics* beschäftigt. Der Fokus lag dabei immer auf dem Einsatz von innovativen Methoden in der Eigenfertigungsplanung und -steuerung der *Frontend* (FE) Standorte der Infineon Technologies AG. Die Beispiele dieser Arbeit entspringen daher dem Kontext der Halbleiterindustrie.

Gleichzeitig war der Autor in das interdisziplinäre Ausbildungsprogramm der GSaME eingebunden. Durch die Einbindung in eine leistungsstarke Forschungsumgebung der GSaME und den regelmäßigen Austausch mit Kollegen der GSaME, der Universität Stuttgart und des Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA[3] wurde die wissenschaftliche Qualität der Arbeit garantiert. Dank der Angebote der GSaME, insbesondere Industriebesuche und Vorträge aus der Industrie, erlangte der Autor tiefe Einblicke in weitere Industriezweige, vor allem in die Automobilindustrie und den Maschinenbau.

Die Arbeiten des Grundlagenbereichs werden in den Kapiteln 3 und 5 besprochen. Zwei Werke von Mnih stechen dabei heraus: In Mnih et al. (2013) und Mnih et al. (2015) wird der *Deep-Q-Network (DQN)*-Agent eingeführt, der die Grundlage der Methodik dieser Arbeit bildet. Insbesondere Mnih et al. (2015) war für den Autor dieser Arbeit in den Überlegungen zu ML prägend und beeinflusste entscheidend die Konzeption des RL-Systems.

[1] Graduate School of Excellence advanced Manufacturing Engineering in Stuttgart (GSaME), Universität Stuttgart, Nobelstr. 12, 70569 Stuttgart, Deutschland
[2] Infineon Technologies AG, Am Campeon 1–15, 85579 Neubiberg, Deutschland
[3] Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Nobelstr. 12, 70569 Stuttgart, Deutschland

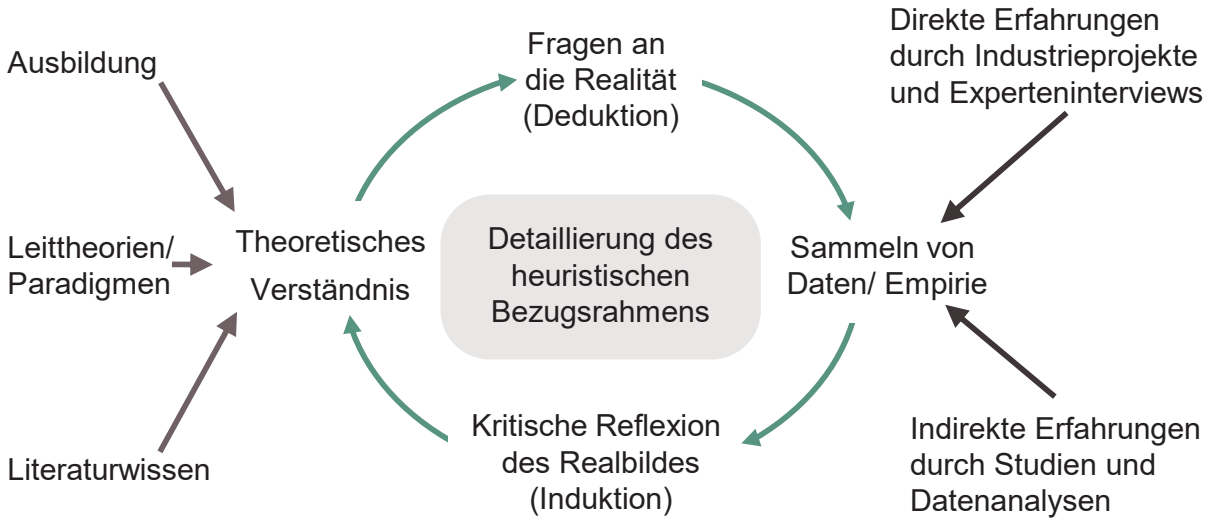


Abbildung 1.6.: Forschungsmethodisches Vorgehen. Basierend auf Kubicek (1976, S. 13)

1.4.3. Begründungszusammenhang

Beobachtungen und Experimente sind primär singuläre Ereignisse. Für die Formulierung einer wissenschaftlichen Aussage ist eine Verallgemeinerung notwendig, die eine induktive Schlussfolgerung voraussetzt (Ulrich et al., 1976b).

Auf der anderen Seite müssen theoretische Aussagen durch gezielte Fragen an die Realität empirisch überprüft werden. Beide Seiten werden vom Begründungszusammenhang abgedeckt: Die empirische Überprüfung des gedanklichen Bezugsrahmens wird in einer Methodik abgebildet (Ulrich et al., 1976a). Dafür ist das Wahrheitskriterium notwendig, also die Konsistenz der Aussagen mit der Wirklichkeit.

Induktives und deduktives Vorgehen in der Wissenschaft sind in Abbildung 1.6 nach Kubicek (1976, S. 13) dargestellt. Dabei handelt es sich um einen hermeneutischen Zirkel, in dem das theoretische Verständnis durch den regelmäßigen Abgleich mit der Empirie kontinuierlich verfeinert wird.

Popper (1973, S. 6 ff.; 2013, S. 12 ff.) hinterfragt dabei den induktiven Schluss von Aussagen. Wissenschaftliche Aussagen können nach Popper nicht verifiziert, sondern nur falsifiziert werden, da die Singularität der

empirischen Beobachtung nicht aufgehoben werden kann. Damit bleibt das Induktionsproblem logisch unlösbar. Popper betrachtet daher in seinem *Kritischen Rationalismus (KR)* die deduktive Methode als einzige wissenschaftliche Vorgehensweise.

In der Arbeit wird anerkannt, dass der wissenschaftliche Forschungsprozess meistens mit einer Idee oder einem Gedanken aus dem eigenen Erfahrungsschatz beginnt. Popper et al. (2013, S. 5 f.) spricht von einer „unbegründete[n] Antizipation" oder „Intuition", die einer Entstehungslogik entbehren. Trotz des induktiven Anfangs des Forschungsprozesses nutzt diese Arbeit die deduktive Methodik für die Überprüfung ihrer Aussagen. Dabei wird insbesondere bei der Formulierung wissenschaftlicher Hypothesen darauf geachtet, dass diese stets falsifizierbar sind.

Lakatos (1982) gibt zu bedenken, dass die Falsifikation immer an eine Reihe von Hilfshypothesen, Paradigmen, Messmethoden, usw. gekoppelt ist. Ein Beispiel dafür ist das „Bohr'sche Atommodell", das aufgrund alter, falscher Paradigmen zuerst „erfolgreich" falsifiziert wurde, bevor es zu einem wissenschaftlichen Standard wurde. Lakatos (1982) erweitert damit den Begriff von Kuhns Paradigmen zu miteinander konkurrierenden Forschungsprogrammen, die aus Theoriesystemen und Methodenregeln bestehen. Wissenschaftliche Aussagen sind innerhalb eines Forschungsprogramms gültig bzw. falsifizierbar.

In dieser Arbeit ist eine wissenschaftliche Hypothese eine im Rahmen des Forschungsprogramms falsifizierbare, objektive Aussage, die aufgrund einer möglichst geringen Anzahl an Annahmen überprüfbare Aussagen über den Forschungsgegenstand macht.

1.4.4. Verwendungszusammenhang

Wissenschaften haben den grundlegenden Anspruch der gesellschaftlichen Relevanz (Ulrich et al., 1976a). Dabei geht es um Zweck und Verwendung der wissenschaftlichen Ergebnisse, die sich damit einem Nutzenkriterium unterwerfen müssen.

Die Methodik dieser Arbeit wurde im Rahmen eines Kooperationsprojekts mit der Industrie entwickelt. Damit ist sie direkt relevant für die Produktivitätssteigerung in der Halbleiterindustrie. Die Digitalisierung und der Einsatz von innovativen Methoden in der Produktion hat jedoch auch gesellschaftliche Auswirkungen. Die Produktion sowie die damit verbundenen Arbeitsplätze können durch Innovation und gesteigerte Wettbewerbsfähigkeit in Hochlohnländern gehalten werden, was langfristigen Wohlstand sicherstellt.

1.5. Forschungsprozess und Aufbau der Arbeit

Der Aufbau der Arbeit ergibt sich aus dem wissenschaftlichen Forschungsprozess. Dabei folgt die Arbeit den Phasen der angewandten Forschung nach Ulrich et al. (2001, S. 195). In Abbildung 1.7 ist die Verbindung der Forschungsphasen mit den strukturierenden Kapiteln der Arbeit aufgezeigt.

Die Arbeit beginnt in Kapitel 1 mit der Ausgangssituation und Problemstellung. Davon abgeleitet sind die Zielsetzung und die Forschungsfrage, die in der Arbeit beantwortet wird. Abschließend wird das wissenschaftliche Vorgehen der Arbeit festgelegt.

Im nächsten Teil, Kapitel 2, werden die Grundlagen für die folgenden Kapitel gelegt. Dabei wird auf Produktionsprinzipien, insbesondere die komplexe Werkstattfertigung sowie auf die Eigenfertigungsplanung und -steuerung eingegangen.

Die Arbeit baut auf den Methoden des ML auf. Die Grundlagen und relevanten Verfahren des Bereichs, die später in der entwickelten Methodik genutzt werden, stellt Kapitel 3 vor.

Die Forschung wurde durch eine Problemstellung aus der Halbleiterindustrie motiviert. Anschließend an die Grundlagen wird der Stand der Technik anhand eines Beispiels aus der Halbleiterindustrie in Kapitel 4 beleuchtet.

Nachdem die konkrete Problemstellung dargelegt ist, werden in Kapitel 5 Anforderungen an eine mögliche Lösung formuliert. Bestehende Lösungs-

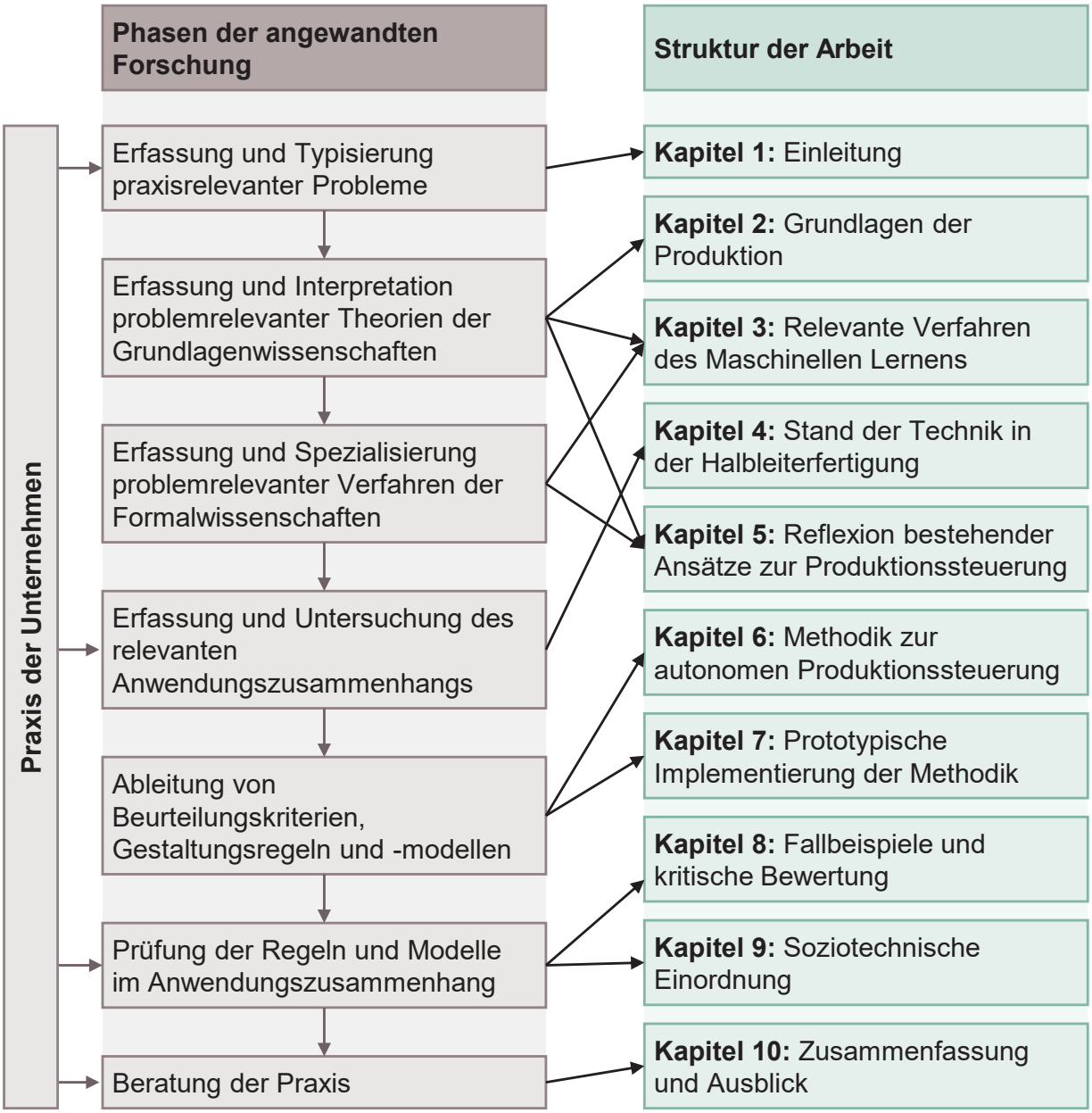


Abbildung 1.7.: Forschungsphasen nach Ulrich und Struktur der Arbeit. Basierend auf (Ulrich et al., 2001, S. 195)

ansätze werden in Bezug auf diese Anforderungen bewertet. Aus dieser Defizitbewertung ergibt sich ein klarer Handlungsbedarf.

In Kapitel 6 wird der Lösungsansatz der Arbeit vorgestellt. Danach wird in Kapitel 7 die prototypische Implementierung in der Halbleiterindustrie beschrieben.

Zur Validierung in Kapitel 8 werden zwei Fallbeispiele aus der Halbleiterindustrie herangezogen. In diesen konnte die Methodik in einer kritischen Reflexion bestätigt werden.

Die soziotechnische Einordnung in Kapitel 9 rundet die Arbeit ab. Im Kapitel wird diskutiert, wie Werker die Abarbeitungsreihenfolge umsetzen, welche Besonderheiten dabei zu beachten sind und welche Konsequenzen dies für die Methodik der vorliegenden Arbeit hat.

Die Arbeit endet mit der Zusammenfassung und einem Ausblick in Kapitel 10.

Kapitel 2

Grundlagen der Produktion: komplexe Werkstattfertigung

Die Arbeit beschäftigt sich mit der autonomen Entscheidungsfindung in der Produktionssteuerung komplexer Werkstattfertigungen. In diesem Kapitel werden die Grundlagen der Produktion in Bezug auf den Titel der Arbeit definiert. Dabei wird insbesondere der Unterschied zwischen „automatisch“ und „autonom“ in Kapitel 2.1 erklärt. Danach wird in Kapitel 2.2 das Fertigungsprinzip, die komplexe Werkstattfertigung, eingeordnet und genauer charakterisiert. Die Ziele und der Ablauf der Eigenfertigungsplanung und -steuerung einer komplexen Werkstattfertigung werden in Kapitel 2.3 erläutert. Zuletzt wird in Kapitel 2.4 auf logistische Kennlinien eingegangen, die für die Charakterisierung des Betriebspunktes und das Management einer komplexen Werkstattfertigung von entscheidender Bedeutung sind.

2.1. Definition und Abgrenzung von automatischen und autonomen Systemen

Im Folgenden wird ein autonomes System von einem automatischen abgegrenzt. Das Adjektiv „automatisch“ beschreibt einen Vorgang der „durch Selbststeuerung oder -regelung [erfolgt]“ (Dudenredaktion, 2017). Das Gabler Wirtschaftslexikon beschreibt „Automatisierung“ als „Übertragung von Funktionen des Produktionsprozesses, insbesondere Prozesssteuerungs und -regelungsaufgaben vom Menschen auf künstliche Systeme“ (Voigt, 2015). Ein automatisches System steuert nach Regeln, die dem System vom Menschen vorgegeben wurden und somit als fest zu betrachten sind.

Der Duden definiert „autonom“ als „selbstständig, unabhängig, [...], eigenständig“ (Dudenredaktion, 2018a). Im Unterschied zum automatischen System basiert das autonome System nicht auf einem starren Regelsystem, das abhängig vom Zustand ausgeführt wird. Die Handlungsstrategien des autonomen Systems werden nicht vom Menschen definiert sondern vom System selbst gefunden. Das autonome System kann dabei von der als optimal bekannten Strategie abweichen, um neue Strategien zu erkunden.

In der Ausführung der erlernten Regeln ist der Übergang zwischen „automatisch“ und „autonom“ fließend. In dieser Arbeit wird die Abgrenzung verwendet, dass autonome Systeme ihre Strategien durch Beobachtung und Exploration selbstständig erlernen, während automatische Systeme einen vorliegenden, festen Regelsatz ausführen.

2.2. Fertigungsprinzipien: Einordnung der komplexen Werkstattfertigung

Die hauptsächlichen Produktionsprinzipien nach der Anordnung der Arbeitssysteme und dem organisatorischen Aufbau sind die Werkstattproduktion, die Gruppenfertigung und die Fließfertigung (Voigt et al., 2017). Wie in Tabelle 2.1 gezeigt, eignen sich die Produktionstypen für unterschiedliche Stückzahlcharakteristiken. Das Produktionsprinzip *Werkstattfertigung* ist

	Werkstatt-fertigung	Gruppen-fertigung	Fließ-fertigung
Einmalproduktion	**X**	**X**	
Wiederholproduktion	**X**	**X**	
Variantenproduktion		**X**	**(X)**
Serienproduktion		**(X)**	**X**
Massenproduktion			**X**

Tabelle 2.1.: Grundsätzliche Eignung unterschiedlicher Fertigungsprinzipien für die Produktionsarten. Basierend auf Bauernhansl (2015)

prinzipiell für die Produktionsarten *Einmalproduktion* und *Wiederholproduktion* gut geeignet.

Bei der Werkstattfertigung werden gleichartige Produktionseinrichtungen zusammengefasst und bilden abgeschlossene Teilbetriebe (Voigt, 2010). Produkte mit unterschiedlichen Arbeitsplänen nehmen in der Werkstatt unterschiedliche Durchlaufwege.

In der flexiblen Werkstattfertigung können Prozesse alternativ von mehreren Maschinen durchgeführt werden. Meistens bedeutet dies, dass identische Maschinen parallel arbeiten (Mönch et al., 2013, S. 25).

Im Rahmen der kundenindividuellen Massenproduktion kommt der Werkstattfertigung eine neue Bedeutung zu. So erschwert die Variantenvielfalt beispielsweise zunehmend die Austaktung eines Fließbands, weshalb neue Konzepte untersucht werden. Die Werkstattfertigung rückt hier in den Fokus für die Massenproduktion (Kern et al., 2015). Beispiele für diese Entwicklung sind die Automobilindustrie (Kern et al., 2015), aber auch die Halbleiterindustrie (Waschneck et al., 2016a).

Durch die Massenproduktion entstehen neue, emergente Phänomene in der Werkstattfertigung. Die Erweiterung der flexiblen Werkstattfertigung für variable Massenproduktion wird als **komplexe Werkstattproduktion bzw. -fertigung** charakterisiert und im Folgenden vorgestellt (Mason et al., 2002;

Mönch et al., 2013, S. 25). Der Eignungsbereich der Werkstattfertigung erweitert sich dabei in Bezug auf Tabelle 2.1 durch die komplexe Werkstattfertigung bis hin zur Massenproduktion.

Für die vollständige Charakterisierung der komplexen Werkstattfertigung werden zuerst Emergenz und Komplexität definiert. Komplexe Systeme bestehen aus einer Vielzahl von Elementen, die dynamisch interagieren (Bauernhansl, 2014; Holland, 2014, S. 2 ff.). Ihr Verhalten ist nicht exakt berechenbar und prognostizierbar (Bauernhansl, 2014).

Emergenz (lateinisch *emergere* „auftauchen", „herauskommen", „emporsteigen") bezeichnet die Entstehung neuer Eigenschaften, Muster und Strukturen, die auf einer makroskopischen Ebene erscheinen, jedoch nicht den einzelnen, mikroskopischen Teilen des Systems zuzuordnen sind (Goldstein, 1999). Emergente Phänomene treten im Selbstorganisationsprozess komplexer Systeme auf (Goldstein, 1999).

In einer komplexen Werkstattfertigung befinden sich eine Vielzahl von Aufträgen parallel in der Produktion. Die Aufträge konkurrieren um die gleichen Ressourcen in der Produktion, die zur Fertigstellung notwendig sind. Durch stochastische Elemente wie Maschinenausfälle und variierende Prozesszeiten ist das Verhalten des Umlaufbestands nicht deterministisch. Die Auftragsdichte in der Produktion wird zu einem übergeordneten Parameter in der Steuerung, der sich erst durch die hohe Anzahl an Aufträgen bestimmen lässt. Dabei verhält sie sich ähnlich einem Fluss, der Engstellen passieren muss. Dabei kommt es zu emergenten Phänomenen wie Wellen im Umlaufbestand.

> **Beispiel: Welle im Umlaufbestand in einer Halbleiterscheibenfertigung**
>
> Wellen sind ein typisches Beispiel für emergente Phänomene, die nur im Makrosystem zu beobachten sind, jedoch keine Eigenschaft der mikroskopischen Bestandteile sind. In der Produktion werden Wellen im Umlaufbestand beobachtet, die sich über Wochen in der Produktion halten.

Im Folgenden wird ein Beispiel aus der Scheibenfertigung Regensburg der Infineon Technologies AG beschrieben. In den Wochen 14 und 15 des fiskalen Kalenders 2017 stand aus unterschiedlichen Gründen an der heißen Nassätzung mit einem Phosphat nur eine um etwa 20 % reduzierte Kapazität zur Verfügung. Dies betraf insbesondere eine spezifische Technologieklasse *TK_A* mit einer DZ von 13 Wochen.

Durch die reduzierte Kapazität kommt es im Arbeitsplan in den Arbeitsschritten, die für Produktionswoche 3 und 8 vorgesehen sind, zu erheblichen Verzögerungen. In Abbildung 2.1 zeigt sich dieser Effekt deutlich: Über Wochen staut sich Material auf. Gleichzeitig kommt es bei den nachfolgenden Arbeitsschritten zu einer Unterauslastung.

Dieser Effekt setzt sich auch nach der Wiederherstellung der Kapazität fort. Noch Wochen später ist eine Über- und Unterauslastung im Fertigstellungsgrad zu sehen.

Mönch et al. (2013, S. 25) bzw. Mason et al. (2002) listen für die formale Definition der komplexen Werkstattfertigung Eigenschaften auf, die in dieser Arbeit in zwei Kategorien eingeteilt werden:

- **Technologische Restriktionen:** Zeitkopplungen zwischen Einzelprozessanweisungen (EPAs) bei chemischen Prozessen, sequenzabhängige Rüstzeiten, ungleichmäßige Prozesszeiten der unterschiedlichen Aufträge und Schwankungen der Prozesszeit einer EPA, verschiedene Prozesstypen (z.B. Batch-Prozesse mit größeren Losgrößen)
- **Logistische Restriktionen:** Schleifenbildung durch wiedereintretende Lose, Maschinenausfälle und andere Störungen der Produktion, vorgegebene Fälligkeitstermine, unterschiedliche Losgrößen in der Produktion (insbesondere durch Entwicklungslose)

Technologische Restriktionen beziehen sich dabei auf Randbedingungen, deren Ursprung in der technologischen Komplexität des Fertigungsprozesses liegt. Logistische Restriktionen entstehen durch die kostenoptimierte Organisation der Produktion. Die Bedingungen für eine komplexe Werkstattfertigung sind schematisch in Abbildung 2.2 zusammengefasst.

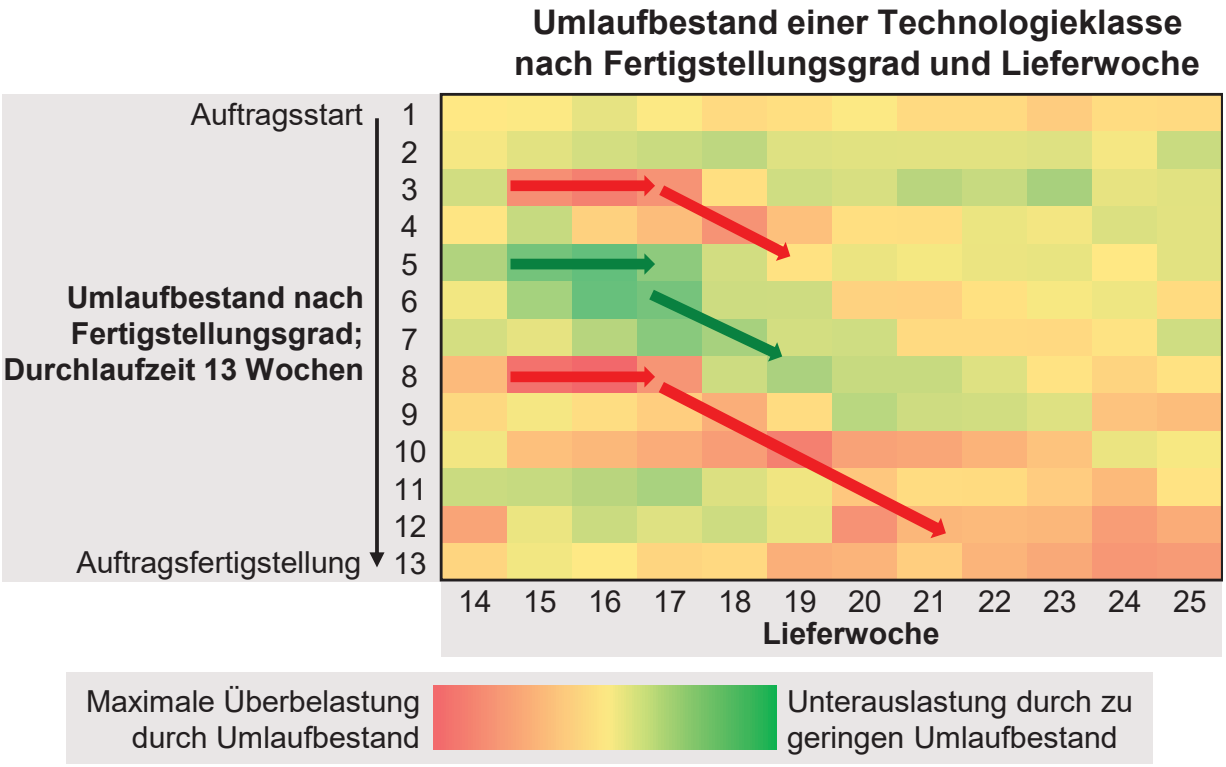


Abbildung 2.1.: Wellen im Umlaufbestand in der Halbleiterfertigung. Basierend auf Daten der Infineon Technologies AG

Dazu kommt die Produktionsquantität der *Massenproduktion*, die eine komplexe Werkstattfertigung charakterisiert. Diese Eigenschaft wird in Mönch et al. (2013, S. 23) nicht explizit in Bezug auf komplexe Werkstattfertigungen genannt, aber implizit durch den Bezug und die Beschreibung der Halbleiterfertigung:

- **Produktionsquantität:** Hervorgerufen durch die Massenproduktion treten emergente Phänomene auf, die ein Resultat der Interaktion der Aufträge in der Produktion sind.

Durch die technologischen und logistischen Restriktionen sowie die hohe Produktquantität wird eine flexible Werkstattfertigung zu einer komplexen.

> **Beispiel: Komplexe Werkstattfertigung in der Halbleiterproduktion**
> Die Halbleiterproduktion ist ein typisches Beispiel für eine komplexe Werkstattfertigung. Durch den hohen Kapitalbedarf für Anlagen liegt

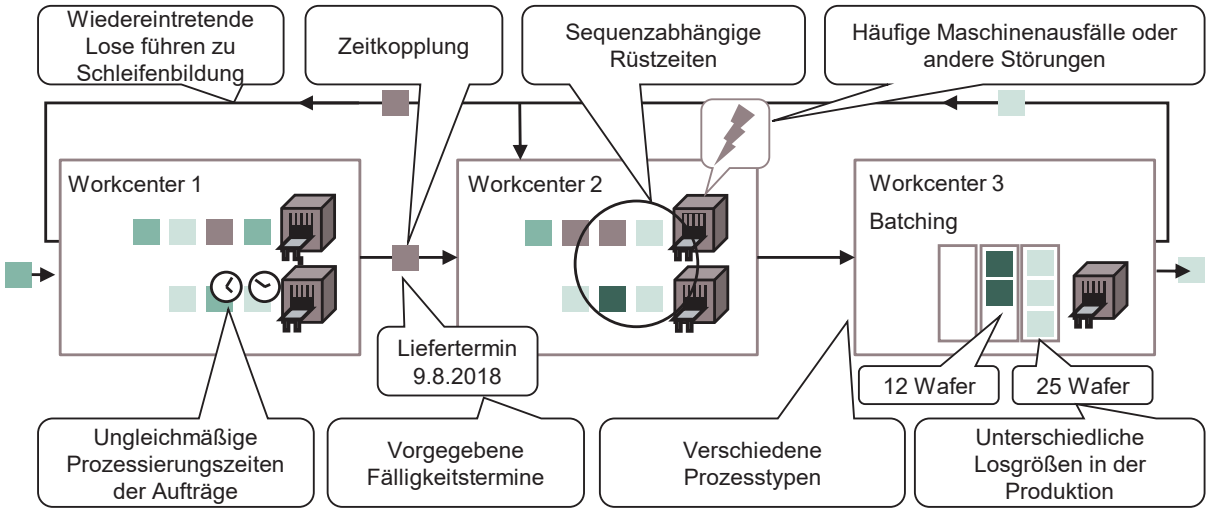


Abbildung 2.2.: Schematische Darstellung der Merkmale einer komplexen Werkstattfertigung. Basierend auf Mason et al. (2002) und Mönch et al. (2013, S. 25)

der Fokus auf einer hohen Auslastung, was zu einem 24/7 Betrieb über 365 Tage im Jahr führt. Zwei Eigenschaften tragen besonders zum komplexen Charakter der Fertigung bei:

- Bei 3*D-NAND-Flash*-Speicher wird bis zu 96 Mal ein ähnlicher Prozessfluss durchlaufen (Luo et al., 2018, 96 Lithographie-Ebenen). Wiedereintretende Lose durchlaufen Schleifen und führen leicht zu einem Aufschaukeln und Verstärken von Problemen in der Produktion.
- Technologie- und Produktentwicklung finden direkt in operativen Produktionslinien statt. Durch die Vermischung von Forschung & Entwicklung (F&E) und operativem Geschäft in der Produktion steigt die Variabilität, beispielsweise von Prozesstypen.

Systematisierung der Werkstattfertigung

Eine Werkstattproduktion ist grundsätzlich in der Lage, aufgrund der verfügbaren Ressourcen M eine Menge an Einzelprozessen durchzuführen. Eine EPA bezeichnet dabei einen Arbeitsschritt an einer Maschine oder an

einem zusammengefassten Maschinencluster. Die Menge aller EPAs wird mit $set(\text{EPA}) = \{\text{EPA}_1, \ldots, \text{EPA}_N\}$ bezeichnet.

Jedes Produkt P, das in der Produktion gefertigt werden kann, hat einen eindeutigen Arbeitsplan. Ein Arbeitsplan R ist eine geordnete Sequenz von EPAs. Jede EPA kann auf einer Untermenge der Maschinen M gefertigt werden. Die Dedizierungsmatrix $d_{m,n}$ legt die Zuordnung von EPAs zu Maschinen fest:

$$d_{m,n} = \begin{cases} 1 & \text{falls Ressource } \boldsymbol{m} \text{ die EPA}_{\boldsymbol{n}} \text{ ausführen kann,} \\ 0 & \text{falls Ressource } \boldsymbol{m} \text{ die EPA}_{\boldsymbol{n}} \text{ nicht ausführen kann.} \end{cases} \tag{2.1}$$

Die Dedizierungsmatrix ist entscheidend für die verfügbare Flexibilität in der Produktion. Jeder Einzelprozessanweisung EPA_n ist eine Rohprozesszeit (RPZ) RPZ_n zugeordnet, die angibt, wie lange die reine Prozesszeit ohne Wartezeiten ist.

2.3. Eigenfertigungsplanung und -steuerung[1]

Die Eigenfertigungsplanung und -steuerung zielt auf die „Erstellung, Umsetzung und Kontrolle eines detaillierten Ablaufplans für die Fertigung“ (Schuh et al., 2014, S. 197) ab, der unter Ausnutzung der tatsächlich verfügbaren Produktionsressourcen den Bedarf bestmöglich abdeckt. Dabei müssen unterschiedliche Zielgrößen der Logistikleistung und der Logistikkosten beachtet werden (Lödding, 2016, S. 21 ff.). Die Zielgrößen lassen sich in externe und interne Faktoren einteilen, die allerdings miteinander verbunden sind (Lödding, 2016, S. 21 ff.). Externe Zielgrößen beeinflussen die Kundenzufriedenheit direkt, während interne Zielgrößen die interne Komplexität und die internen Kosten beeinflussen. Die Zielgrößen der Logistikleistung und der Logistikkosten sind in Tabelle 2.2 zusammengefasst.

Der Planungsprozess umfasst dabei fünf Schritte, die im Folgenden betrachtet werden. Abbildung 2.3 gibt einen Überblick über den Prozess. Bei der Losgrößenrechnung muss ein Kompromiss zwischen höheren Beständen

[1] Das Kapitel basiert auf der Grundlagenliteratur Schuh et al. (2014, S. 197 ff.).

	Logistikleistung	Logistikkosten
extern	*Auftragsfertigung* Lieferzeit Lieferterminabweichung Liefertreue	Preis
extern	*Lagerfertigung* Servicegrad	
intern	Durchlaufzeit Terminabweichung Termintreue	Bestand Auslastung Verzugskosten

Tabelle 2.2.: Zielgrößen der Logistikleistung und der Logistikkosten. Nach Lödding (2016, S. 21 ff.)

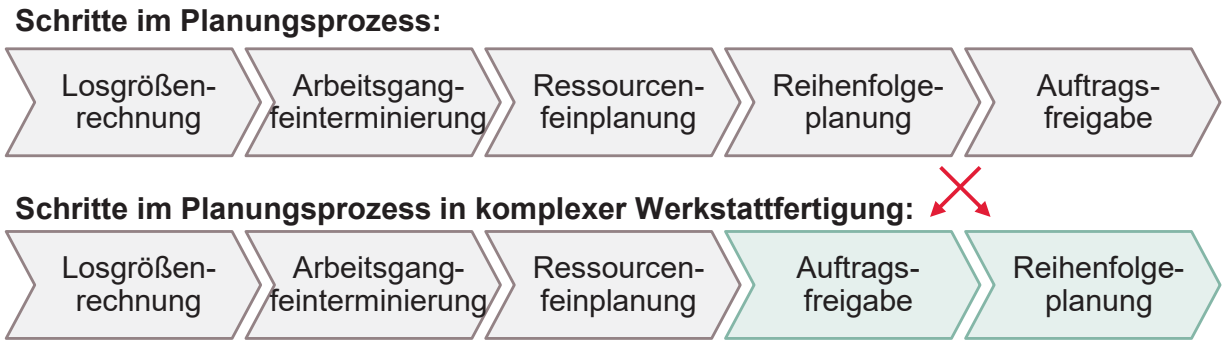


Abbildung 2.3.: Oben: Eigenfertigungsplanung und -steuerung nach Schuh et al. (2014, S. 199). Unten: Eigenfertigungsplanung und -steuerung in einer komplexen Werkstattfertigung

durch größere Losgrößen und höheren Rüstkosten bei kleineren, wechselnden Losgrößen gefunden werden. Ziel ist es, die wirtschaftlich optimale Größe für die Produktion zu bestimmen. Im nächsten Schritt, der Arbeitsgangfeinterminierung, werden die Start- und Endtermine der Arbeitsgänge bestimmt. Grundlage dafür bildet der Arbeitsplan. Der letzte Endtermin muss so bestimmt werden, dass der Liefertermin eingehalten werden kann. Bei der Arbeitsgangfeinterminierung werden auch Rüst- und Übergangszeiten berücksichtigt.

In den vorherigen Schritten wurde von einer unbegrenzten Kapazität der Ressourcen ausgegangen. Bei der Ressourcenfeinplanung wird der Plan an die tatsächlich vorhanden Kapazitäten angepasst. Dabei werden Überlastungen und Unterauslastungen der Produktionskapazitäten sichtbar. Ziel ist eine gleichmäßige Kapazitätsbelastung der Ressourcen, um Kosten durch Leerkapazität zu minimieren.

Nach den ersten drei Schritten existieren Start- und Endtermine für die einzelnen Arbeitsschritte der Produkte unter Berücksichtigung der verfügbaren Ressourcen. Nicht berücksichtigt sind die Interaktionen der Aufträge an den Maschinen, also ihre Anordnung in der Warteschlange. Die Bearbeitungsreihenfolge innerhalb der Warteschlagen wird in der Reihenfolgeplanung festgelegt. Die Reihenfolgeplanung beeinflusst sowohl die Produktivität, z.B. durch Rüstzeiten, als auch die Termintreue der Fertigung. Die Reihenfolgeplanung kann nach Selektionskriterien wie Prioritätsregeln oder Kumulationskriterien, wie bei der Rüstzeitminimierung, erfolgen. In Kapitel 5 wird genauer auf die Methoden der Reihenfolgeplanung eingegangen.

Der letzte Schritt in der Eigenfertigungsplanung und -steuerung ist die Auftragsfreigabe. Nach der Verfügbarkeitsprüfung des Loses und der Ressourcen wird der Auftrag gestartet.

In einer komplexen Werkstattfertigung sind die letzten zwei Schritte der klassischen Eigenfertigungsplanung, wie in Abbildung 2.3 gezeigt, vertauscht (Ehm et al., 2011). Bereits Schuh et al. (2014, S. 204) deuten an, dass eine explizite Reihenfolgeplanung nicht immer notwendig ist: „Dabei kann aber ggf. auch auf eine explizite Reihenfolgeplanung verzichtet werden, wenn beispielsweise durch eine organisationsbedingt intensive Kommunikation und einen intensiven Erfahrungseinsatz die Mitarbeiter in Fertigungsinseln selbst über die Abarbeitungsreihenfolge entscheiden."

Nach der Ressourcenfeinplanung erfolgt in der komplexen Werkstattfertigung direkt die Auftragsfreigabe ohne eine explizite Verfügbarkeitsprüfung. Die Reihenfolgeplanung wird dann in Echtzeit dezentral durchgeführt. Echtzeit wird nach der Norm DIN 44300 (*DIN 44300-9:1988-11* 1988) wie folgt definiert: „Unter Echtzeit versteht man den Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit

sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen“ (Scholz, 2005, S. 39). Bei der Reihenfolgeplanung ist wichtig, dass die „vorgegebene Zeitspanne“ so gewählt ist, dass keine Leerstandszeiten auf den Maschinen entstehen. Wenn ein Auftrag auf einer Maschine fertig prozessiert ist, entlädt ein Mitarbeiter die Maschine. Währenddessen entscheidet das System welcher Auftrag als nächstes bearbeitet werden soll. Der Mitarbeiter holt das entsprechende Los und startet den Auftrag auf der Maschine. In der Halbleiterfertigung verfügen viele Maschinen über zwei oder mehr *Load Ports*, an denen die Maschine beladen werden kann. Damit kann während des Ent- und Beladens an einem *Load Port* der Auftrag auf dem zweiten *Load Port* parallel prozessiert werden. Dadurch bleibt die Auslastung der Maschinen hoch, auch wenn beim Be- und Entladen der Maschinen kurze Verzögerungen auftreten. Trotzdem liegen die Verarbeitungszeiten in der Reihenfolgeplanung in der Halbleiterindustrie überlicherweise unter einer Sekunde[1].

Der Grund für die Vertauschung der letzten zwei Schritte der klassischen Eigenfertigungsplanung liegt in der hohen Dynamik und Komplexität der komplexen Werkstattfertigung: Die Basis für langfristige Pläne ändert sich ständig. Maschinenausfälle, veränderte Auftragsprioritäten und weitere stochastische Elemente führen zu sich kontinuierlich verändernden Bedingungen. In der Halbleiterfertigung liegt die dominierende Periodendauer der Veränderungen, die eine Neuplanung notwendig macht, bei fünf Minuten (Govind et al., 2008). Gleichzeitig ist die Komplexität der Planung bei hunderten Maschinen und tausenden Aufträgen mit bis zu 1000 Prozessschritten und DZs von mehreren Wochen enorm hoch.

Der Ausweg aus dem Dynamik-Komplexitäts-Dilemma ist die dezentrale Planung und Steuerung in Echtzeit. Dieses Paradigma wird auch von der Industrie 4.0 als Zukunftsvision vertreten: „Dezentralität ermöglicht eine klare Koordination und macht Komplexität beherrschbar, da die Steuerungs-

[1] Quelle: Fachabteilung *Manufacturing Excellence* der Infineon Technologies AG

aufgabe nicht mehr ausschließlich an einem Ort erfolgen muss“ (Bischoff et al., 2015, S. 17). In der Studie von Bischoff et al. (2015, S. 21) wird „Selbstorganisation & Autonomie“ als letzte Ausbaustufe der Industrie 4.0 betrachtet.

Die dezentrale Steuerung arbeitet ereignisorientiert. Bei einem Ereignis, das eine Entscheidung bezüglich der Abarbeitungsreihenfolge benötigt, wird die Steuerung gestartet. Die vorhandenen Aufträge und verfügbaren Produktionsressourcen an einer Maschinengruppe werden lokal erfasst. Auf dieser Grundlage wird die Reihenfolgeplanung in Echtzeit durchgeführt und der zu produzierende Auftrag wird ausgewählt.

Beispiel: Eigenfertigungsplanung und -steuerung in der Scheibenfertigung am Beispiel des Halbleiterherstellers Infineon Technologies AG[a]

In der Halbleiterfertigung sind alle Produktionsmaschinen und das *Automated Material Handling System (AMHS)* auf eine Losgröße von 25 Wafern optimiert. Kleinere Losgrößen, also z.B. 15 Wafer pro Los, bedeuten häufig, dass Maschinenkapazitäten nicht vollständig ausgeschöpft werden. Da Auslastung aufgrund der kapitalintensiven Produktionsanlagen in der Halbleiterfertigung hohe Priorität hat, werden im operativen Betrieb fast ausschließlich vollständige Lose mit 25 Wafern gestartet. Die Anzahl der Chips pro Wafer ist fest und kann nicht variiert werden. Bei Produkten mit geringerer Umschlagsgeschwindigkeit werden höhere Bestände in Kauf genommen.

Die Arbeitsgangfeinterminierung ist durch den Arbeitsplan der Produkte vorbestimmt. In der Abfolge chemischer, mechanischer und lithographischer Prozesse ist eine Änderung der Reihenfolge nicht möglich. Den Schritten im Arbeitsplan ist jeweils eine RPZ zugeordnet. Die RPZ für jeweils einen Prozessschritt wird mit einem Plan-Flussfaktor (Plan-FF; siehe Kapitel 2.4) multipliziert, um einen Richtwert für Transport-, Warte- und Fertigungszeit zu erhalten. Mit dem Flussfaktor (FF) wird gesteuert, wie schnell bzw. mit welcher Priorität ein Los gefertigt

werden soll. Bei Eillosen kann ein Wert von $FF < 1,5$ erreicht werden, typisch sind eher Werte von $FF = 3$. Ausgehend vom Liefertermin wird nun für jede EPA ein Plandatum errechnet.

Bei der Ressourcenfeinplanung liegt der Fokus auf der Kapazitätsbeschränkung der installierten Anlagen, die im 24/7 Betrieb laufen. Repetierfaktoren, also Werkstoffe die im Produktionsprozess unmittelbar verbraucht werden (Gutenberg, 1951, S. 3 ff.), sind für die meisten Halbleiterprodukte identisch und werden in großen Mengen – bei Gasen teilweise in einem Vorrat für mehrere Monate – eingekauft. Die Anzahl der Werker wird immer an den verfügbaren Anlagen ausgerichtet, sodass Anlagen nie wegen einem Mangel an Werkern stillstehen. Der entscheidende Teil der Planung liegt also beim Abgleich des Produktionsprogramms mit der installierten Anlagenkapazität. Über die Dedizierungsmatrix werden Anlangen in Maschinengruppen, sogenannte *Ressourcenpools*, eingeteilt. Über mathematische Optimierung wird dann das Produktionsprogramm angepasst. Dabei kann es sein, dass Produkte, die erst für die nächsten Wochen geplant sind, vorgezogen werden, um den Maschinenpark vollständig auszulasten.

Danach erfolgt direkt die Auftragsfreigabe: Für jeden Auftrag wird ein neues Los in den Reinraum eingeschleust und der Produktionsprozess beginnt.

Die DZs betragen um die 100 Tage mit bis zu 1000 Prozessschritten pro Produkt. Gleichzeitig befinden sich über 100.000 Wafer in einer Linie, auf denen über 1000 unterschiedliche Produkte gefertigt werden. Bei dieser Komplexität erfolgt die Reihenfolgeplanung in Echtzeit an den Maschinen. Meistens wird anhand einer Hierarchie von Heuristiken die Reihenfolge der Warteschlange an einer Maschinengruppe bestimmt. Genauer wird auf diesen Planungsschritt noch in Kapitel 4.2 eingegangen.

[a] Basierend auf Expertengesprächen und der eigenen Tätigkeit in Projekten im Unternehmen Infineon Technologies AG

Bei der Reihenfolgeplanung in Echtzeit wird weiter zwischen *Scheduling* und *Dispatching* unterschieden. *Scheduling* bezieht sich auf die Zuordnung von Fertigungsaufträgen zu Ressourcen bzw. Anlagen für einen **bestimmten, zukünftigen Zeitraum** (Aytug et al., 2005). Dabei handelt es sich eigentlich um eine Planung mit geringem Planungshorizont. Für *Scheduling* wird meistens mathematische Optimierung verwendet. Dabei werden Rechenzeiten in Kauf genommen, die eine Echtzeitfähigkeit verhindern.

Die eigentliche **Steuerung in Echtzeit** wird vom *Dispatching*-System übernommen. *Dispatching* beschreibt die Auswahl des nächsten Auftrages aus einer Gruppe von wartenden Aufträgen aufgrund zugeordneter Prioritäten (Mönch et al., 2013, S. 27; Blackstone et al., 1982). Die Prioritäten der Aufträge können manuell, über Heuristiken oder durch das *Scheduling*-System festgelegt werden. Meistens werden die Aufträge über den Bedarf der *Supply Chain (SC)* bzw. manuell in übergeordnete Prioritätsklassen eingeordnet. Die lokale Priorität eines einzelnen Auftrags an einer Anlage wird dann über Heuristiken bestimmt. Sie kann auch vom Anlagenzustand abhängen, beispielsweise bei Rüstzeitoptimierung. Die *Dispatching*-Liste an der Anlage wird dann durch Anordnung der Lose nach absteigender Priorität erstellt.

In der Praxis werden in Halbleiterfertigungen vorwiegend nur *Dispatching*-Heuristiken eingesetzt (Mönch et al., 2013, S. vi). *Scheduling* findet erst in den letzten Jahren den Weg in die Fabriken (Mönch et al., 2013, S. vi), in erster Linie als Hybridverfahren in Kombination mit *Dispatching*.

2.4. Logistische Kerngrößen, Zusammenhänge und Betriebskennlinie in der Werkstattfertigung

Im Folgenden werden FF und Auslastung für diese Arbeit definiert und der mathematische Zusammenhang zwischen beiden, die Betriebskennlinie (engl. *Operating Curve (OC)*), aufgestellt.

Der FF, teilweise auch als Flussgrad (Nyhuis et al., 2012, S. 44) oder *X-Factor* (Ozawa et al., 1999) bezeichnet, kann für einzelne Aufträge, Maschinen und Fabriken berechnet werden.

Der FF eines Auftrags a ist definiert als Division der DZ DZ_a durch die Summe der RPZs:

$$FF_a = \frac{DZ_a}{\sum_{\text{Arbeitsplan } a} RPZ}. \tag{2.2}$$

Wird der FF für eine EPA berechnet, wird dementsprechend die EPA DZ und EPA RPZ verwendet. Gleiches gilt für die Berechnung an einer Maschine oder Fabrik, bei der jeweils die mittlere DZ an der Maschine oder in der Fabrik und die mittlere RPZ an der Maschine bzw. mittlere RPZ der Arbeitspläne benutzt wird.

Für die Auslastung müssen zuerst die möglichen Maschinenzustände festgelegt werden. Diese Arbeit folgt hierbei den Standards *SEMI E10-0304(E)* (2004) und *SEMI E79-1106* (2006), die in der Halbleiterindustrie Anwendung finden. Die einzelnen Maschinenzustände sind in Abbildung 2.4 aufgeführt.

Für die Definition der *Overall Equipment Efficiency (OEE)*[1] muss zusätzlich zu den Maschinenzuständen die Ausbeute-Effizienz (engl. *Yield Efficiency*, auch als *Quality Efficiency* bezeichnet) definiert werden, die den Anteil der einwandfreien Bauteile für den Verkauf an den insgesamt produzierten Bauteilen angibt. Damit wird die OEE nach *SEMI E79-1106* (2006) wie folgt definiert:

$$\begin{aligned} OEE &= \textit{(Yield Efficiency)} \cdot \textit{(Availability Efficiency)} \ \ldots \\ &\quad \cdot \ \textit{(Performance Efficiency)} \\ &= \textit{(Yield Efficiency)} \cdot \frac{\textit{(Productive Time)}}{\textit{(Total Time)}}. \end{aligned} \tag{2.3}$$

[1] Die Norm *SEMI E79-1106* (2006) folgt nicht der üblichen Bezeichnung *Overall Equipment **Effectiveness***. In der Norm wird dazu angemerkt: „While traditional total productive maintenance (TPM) literature [...] calculates overall equipment *effectiveness* (also using the same acronym OEE) based on units of production, overall equipment *efficiency* (OEE) is the more correct term for the time-based calculations defined in this specification. Traditional unit-based overall equipment effectiveness calculations and the time-based OEE calculations within this specification yield essentially the same results.“ Dies entspricht nicht der im deutschsprachigen Raum üblichen Unterscheidung von Effektivität und Effizienz.

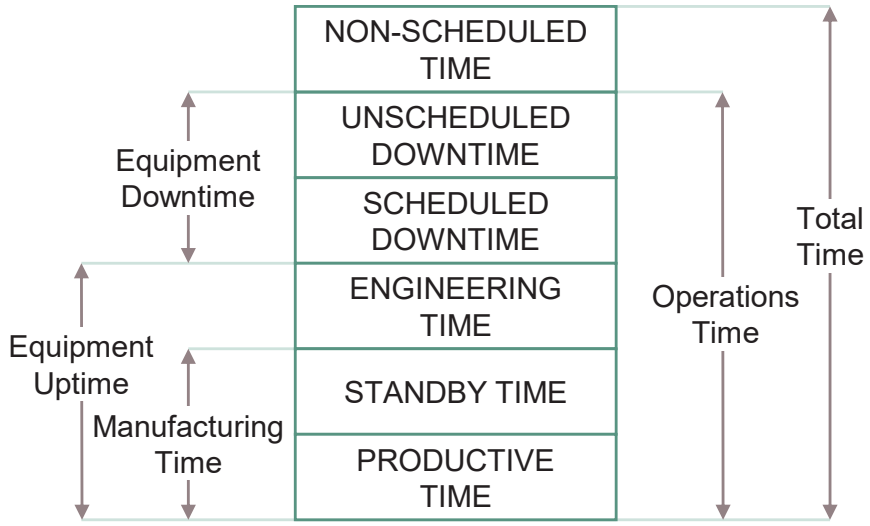


Abbildung 2.4.: Bezeichnung und Einteilung der Maschinenzustände. Basierend auf *SEMI E10-0304(E)* (2004, Figure 1)

In die OEE Betrachtung fließt die Prozessqualität mit ein. Diese kann durch *Dispatching* und *Scheduling* – dem Fokus dieser Arbeit – nicht direkt beeinflusst werden. Daher wird auf Grundlage der *SEMI E10-0304(E)* (2004) die logistische Fertigungsauslastung U (engl. *Manufacturing Uptime Utilization*, auch UU_m) definiert[1]:

$$U = \frac{\textit{Productive Time}}{\textit{Manufacturing Time}}. \tag{2.4}$$

Im Folgenden bezieht sich Auslastung immer auf die logistische Fertigungsauslastung. Die Auslastung misst den Anteil der OEE, der direkt von *Dispatching* und *Scheduling* beeinflusst wird.

Die Wartezeit und damit indirekt der FF einzelner Maschinen, aber auch Maschinengruppen, lässt sich statistisch mit der Warteschlangentheorie beschreiben, die eine Teildisziplin der Mathematik ist. Die Warteschlangentheorie vermittelt ein gutes Verständnis für die theoretischen Zusammenhänge zwischen den Kerngrößen einer Werkstattfertigung.

[1] Die Definition folgt den Technischen Richtlinien *TR25* (2008) und *TR27* (2010) der Infineon Technologies AG. Die Quelle ist nur für Mitarbeiter der Infineon Technologies AG zugänglich.

Das *Gesetz von Little* ist ein universeller Zusammenhang für die Berechnung der Durchlaufzeit eines stationären Systems (Little, 1961; Nyhuis et al., 2012, S. 32):

$$\overline{DZ} = \frac{\overline{WIP}}{\lambda}, \tag{2.5}$$

wobei $\overline{DZ}$ die mittlere DZ und $\overline{WIP}$ den mittleren Umlaufbestand bezeichnen. Der Parameter λ bezieht sich auf die mittlere Ankunftsrate der Aufträge am System, die im stationären Fall identisch mit dem Durchsatz ist. Zur Klassifikation der Modelle wird ein Kennzeichnungssystem eingesetzt (Nyhuis et al., 2012, S. 41), das häufig als Kendall-Notation bezeichnet wird (Hopp et al., 2001, S. 256):

$$A/B/m. \tag{2.6}$$

Dabei bezeichnet A das Ankunftsprozessverhalten und B das Abfertigungsprozessverhalten, die über die Verteilung der Zwischenereigniszeiten charakterisiert sind. Der Parameter m bezieht sich auf die Anzahl der parallelen, identischen Bedienstationen innerhalb des Systems.

Die Warteschlange an einer Maschine lässt sich als $G/G/1$ Warteschlange beschreiben. Der Parameter G beschreibt eine generelle Verteilung, bei der die Zwischenabfertigungszeiten beliebig verteilt und voneinander stochastisch unabhängig sind.

In einem System $G/G/1$ mit hoher Auslastung U lässt sich der Erwartungswert der Wartezeit $\mathbb{E}(W)$ durch die Kingman-Formel (Kingman et al., 2003; Shanthikumar et al., 2007) annähern:

$$\mathbb{E}(W) \approx \left(\frac{c_a^2 + c_s^2}{2} \right) \cdot \left(\frac{U}{1-U} \right) \cdot RPZ, \tag{2.7}$$

wobei c_a^2 den Variationskoeffizient auf den Ankunftszeiten und c_s^2 den Variationskoeffizient auf den Servicezeiten beschreibt. Die Variationskoeffizienten werden häufig zur Variabilität α zusammengefasst:

$$\alpha = \frac{c_a^2 + c_s^2}{2}. \tag{2.8}$$

Werden Gleichung 2.7 und 2.8 in Gleichung 2.2 eingesetzt, ergibt sich die Betriebskennlinie:

$$\begin{aligned} FF &= \frac{DZ}{RPZ} \approx \frac{\mathbb{E}(W) + RPZ}{RPZ} \\ &= \alpha \cdot \left(\frac{U}{1-U} \right) + 1. \end{aligned} \tag{2.9}$$

Die Betriebskennlinie stellt einen Zusammenhang zwischen den logistischen Kerngrößen *Auslastung* und *FF* her. Im industriellen Umfeld wird Gleichung 2.9 auch für die Beschreibung ganzer Fabriken verwendet, wobei dann der FF der Fabrik und die typische Maschinenauslastung verwendet werden.

Für Maschinengruppen mit mehreren identischen Maschinen m mit einer $G/G/m$ Warteschlange existiert ebenfalls eine Näherung (Whitt, 1983; Hopp et al., 2001, S. 273):

$$\mathbb{E}(W) \approx \alpha \cdot \left(\frac{U^{\sqrt{2(m+1)}-1}}{m \cdot (1-U)} \right) \cdot RPZ. \tag{2.10}$$

Diese reduziert sich bei $m = 1$ auf Gleichung 2.7.

Mehrere Betriebskennlinien für $G/G/1$-Systeme für verschiedene Variabilitäten α sind in Abbildung 2.5 gezeigt.

Die Betriebskennlinie verdeutlicht die Austauschbeziehung zwischen hoher Auslastung und einer schnellen DZ, was gleichbedeutend mit einem kleinen FF ist. Der optimale Betriebspunkt auf der Betriebskennlinie wird je nach Positionierung des Unternehmens am Markt bestimmt. Eine kosteneffiziente Fertigung ist nur bei hoher Auslastung möglich, was automatisch zu einem hohen Umlaufbestand und langen DZs führt. Soll dagegen mit kurzen DZs schnell und flexibel am Markt reagiert werden können, führt

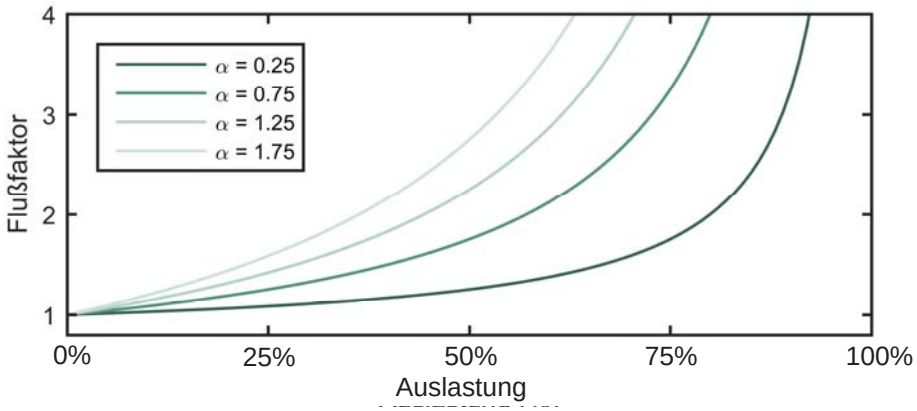


Abbildung 2.5.: Betriebskennlinie nach Gleichung 2.9 für verschiedene α-Werte

dies zu einer schlechteren Auslastung der kapitalintensiven Anlagen. Das Senken der Variabilität α ist in jedem Fall sinnvoll, aber meistens aufgrund des Produktionsportfolios und der Prozesskomplexität nur bedingt möglich.

Kapitel 3

Relevante Verfahren des Maschinellen Lernens

Die Grundlage für die Diskussion um KI legte Alan Turing 1950 in seinem Aufsatz *„Computing Machinery and Intelligence“* (Turing, 2009). Erstmals verwendet wurde der Begriff KI vom amerikanischen Informatiker John McCarthy 1956 in einem Projektantrag für eine Konferenz im *Dartmonth College* in den USA (Wichert, 2000). Das Thema KI schürte in den folgenden Jahrzehnten in regelmäßigen Wellen überzogene Erwartungen und philosophische Diskussionen um die Definition. In dieser Arbeit wird KI definiert als die Erforschung von künstlichen, intelligenten Agenten, die eine Umgebung wahrnehmen und darin zielgerichtete Aktionen ausführen (Russell et al., 2010, S. viii u. 4 f.). Der Agent hat die Fähigkeit, sich in unbekannten Umgebungen zurechtzufinden. Diese Fähigkeit zur Adaption an Unbekanntes bzw. Neues wird als *Lernen* verstanden (Russell et al., 2010, S. viii).

Ein Teilgebiet der KI ist das Maschinelle Lernen (ML). Das ML bezeichnet die Fähigkeit von Computerprogrammen durch Parameteroptimierung, Wissen und Strategien zu erlernen ohne explizit dafür programmiert zu

werden (Koza et al., 1996; Samuel, 1959). Das ML wird in drei Teilbereiche unterteilt, die sich durch die Art der Problemstellung und des Lernens unterscheiden (Silver, 2015):

- ***Supervised Learning* (SL, dt. überwachtes Lernen).** Bei SL wird ein System trainiert, bei einer bestimmten Eingabe eine bestimmte Ausgabe zu erzeugen. Anhand von Beispielen mit Ein- und Ausgaben (Trainingsdaten) stellt das System eine Hypothese auf, wie die Eingabe auf die Ausgangswerte abgebildet werden kann. Unterschieden wird zwischen Regressionsverfahren für kontinuierliche Ausgangswerte und Regressionsverfahren für Klassifikation von diskreten Daten.
- ***Unsupervised Learning* (UL, dt. unüberwachtes Lernen).** Beim UL sollen Muster in Eingangsdaten gefunden werden, wobei dem lernenden System keine Zielwerte oder Belohnungen im Sinne von Beispielen für das Training bekannt sind. Teilweise wird die Anzahl der erwarteten *Cluster* vorgegeben. UL lässt sich in *Clustering* und Dimensionalitätsreduktion unterteilen.
- ***Reinforcement Learning* (RL, dt. bestärkendes Lernen)**. Beim RL lernt ein Agent eine Strategie, mit der er seine Belohnung in einer Trainingsumgebung maximiert. Die Aktionen des Agenten in der Umgebung beeinflussen den folgenden Zustand, den der Agent sieht. Damit ist die Problemstellung immer sequenziell. Das Feedback über die Belohnung kann verzögert erfolgen.

Eine Methode, die in allen Bereichen des MLs eingesetzt werden kann, ist DL. Damit werden Neuronale Netze (NNs, siehe Kapitel 3.1) mit mehreren, häufig spezialisierten, verdeckten Ebenen (engl. *hidden layers*) bezeichnet, die im Gegensatz zu einfachen NNs direkt mit Rohdaten trainiert werden (LeCun et al., 2015). Dagegen benötigen einfache NNs Eingangswerte (engl. *features*), die zunächst von menschlichen Experten gewählt und vorverarbeitet werden. Durch die hohe Anzahl an Ebenen im DL erlernt das NN die *Features*, also die spezifischen Eigenschaften der Eingangswerte, selbst. Der Durchbruch von DL wurde von großen Datenmengen und der Verfügbarkeit

Abbildung 3.1.: Überblick über die vorgestellten Bereiche der KI und Einordnung des autonomen *Dispatching*-Konzepts

von Rechenleistung begünstigt. Die Funktionsweise von NNs und DL wird in Kapitel 3.1 erklärt.

Ein Überblick über die vorgestellten Bereiche und Techniken der KI ist in Abbildung 3.1 gezeigt. Das Konzept der Arbeit liegt im Schnittbereich von RL und DL. In den folgenden Kapiteln 3.1 und 3.2 werden DL und RL näher vorgestellt. Zuletzt wird in Kapitel 3.3 auf die Charakteristiken von Multi-Agenten-Systemen (MAS) und ihren Umgebungen eingegangen.

3.1. Neuronale Netze

Die NNs bestehen aus der Verknüpfung einer großen Zahl relativ einfacher Recheneinheiten, den Neuronen, die gemeinsam eine logistische Regression ausführen. Vorteile eines NNs sind, dass komplexe, nichtlineare Hypothesen nicht explizit formuliert werden müssen und ein großer Raum an Eingangsvariablen verarbeitet werden kann. Die Struktur von NNs orientiert sich an der des menschlichen Gehirns mit Neuronen und Axonen.

Ein neuronales Netz besteht aus einer Eingangsebene, die einen Vektor als Eingabe x hat. Es folgen eine beliebige Anzahl verdeckter Ebenen, die aus Neuronen bestehen, und am Ende folgt eine Ausgabeebene, die wieder einen Vektor zurückgibt. Neuronen müssen nicht unbedingt in Ebenen angeordnet werden. Allerdings erwies sich diese Anordnung in vielen Anwendungen als

außerordentlich erfolgreich. Des Weiteren hat die Anordnung Vorteile bei der Berechnung, da die Ausgaben der Neuronen einer Ebene zusammen in einer Matrix-Operation bestimmt werden können.

In jeder Ebene n werden die Eingangswerte $a^{(n-1)}$ (bei der ersten Ebene $x = a^{(0)}$) mit einer Gewichte-Matrix $\theta^{(n)}$ multipliziert. Das hochgestellte (n) steht dabei für die Ebene; Vektor- und Matrixindices werden wie gewöhnlich unten notiert. Eine Transfer- oder Aktivierungsfunktion g bestimmt dann, welcher Wert an die nächste Ebene als Eingangssignal übermittelt wird:

$$a^{(n)} = g\left(\theta^{(n)} \cdot \ a^{(n-1)}\right). \tag{3.1}$$

Die Transferfunktion wird dabei auf jedes Vektorelement einzeln angewendet. Typische Transferfunktionen sind Sigmoidfunktionen, Tangens hyperbolicus und *Rectified Linear Units (ReLU)*. Die Sigmoidfunktion folgt einer charakteristischen „S“-Kurve:

$$g(a) = \frac{1}{1 + e^{-a}}. \tag{3.2}$$

ReLUs sind wie folgt definiert:

$$g(a) = \max(0, a). \tag{3.3}$$

In der Ausgangsebene wird gerne eine *Softmax* Funktion als Transferfunktion verwendet, da sie eine Normalisierung beinhaltet:

$$g(a)_j = \frac{\exp\left(a_j\right)}{\sum_{k=1}^{K} \exp\left(a_k\right)}, \tag{3.4}$$

wobei K die Anzahl der Neuronen in der Ebene beschreibt.

Ein Beispiel für ein NN mit zwei verdeckten Ebenen ist in Abbildung 3.2 gezeigt. Es klassifiziert die Eingangswerte in zwei diskrete Klassen. Die Ausgangsneuronen geben die Wahrscheinlichkeit an, dass sich die Eingabe in der entsprechenden Klasse befindet.

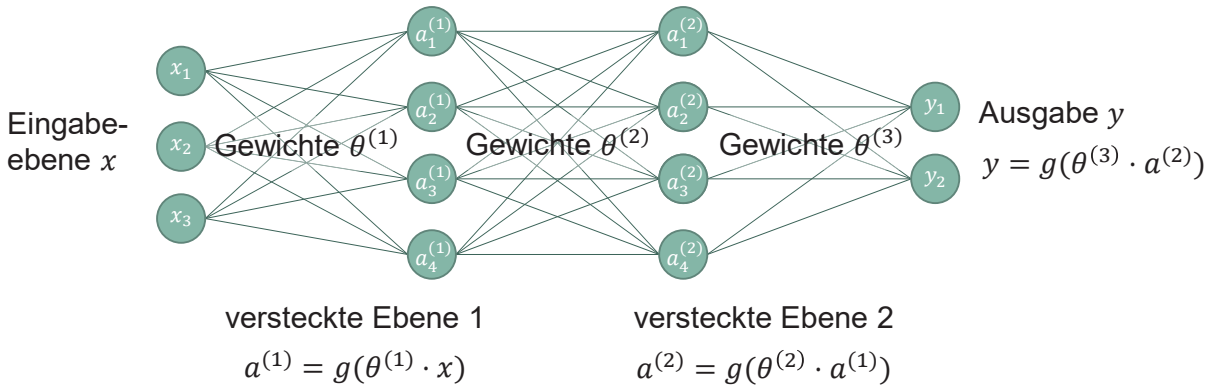


Abbildung 3.2.: Beispiel eines NNs mit zwei verdeckten Ebenen. Das NN bekommt drei Eingangswerte und hat in der Ausgangsebene zwei Neuronen. Diese könnten beispielsweise eine Klassifizierung in zwei Klassen vornehmen.

Es wird zwischen *Feedforward NNs* und *Recurrent NNs (RNNs)* unterschieden. Bei *Feedforward NNs* gibt es im Gegensatz zu den *RNNs* keine Einheiten, die Schleifen bilden. Die Information bewegt sich dadurch immer nur in eine Richtung im Netzwerk.

Tiefe NNs, die im DL eingesetzt werden, zeichnen sich durch ihre hohe Anzahl verdeckter Ebenen sowie weiterer Ebenentypen wie z.B. *Convolutional Layer* aus.

Die NNs werden trainiert, indem die Gewichte so angepasst werden, dass bei bestimmten Eingangsdaten gewünschte Ausgangswerte berechnet werden. Für diese Anpassung werden stochastische Gradientenverfahren (engl. *Stochastic Gradient Descent (SGD)*) eingesetzt. Häufig verwendet werden *RMSProp* (Tieleman et al., 2012) und das *Adam* Verfahren (Kingma et al., 2015), die beide Erweiterungen des klassischen SGD darstellen.

3.2. *Reinforcement Learning*

Hinter RL steht die Grundidee, dass ein Programm durch Interaktion mit einer Umgebung und die gelegentliche Gabe von Belohnungen erlernt, wie es sich in der Umgebung verhalten soll. Ziel des Algorithmus ist die Maximie-

rung der Belohnung. Das RL ist stark inspiriert von menschlichem Lernen, vornehmlich von Kleinkindern.

Das RL rückte insbesondere durch die Erfolge der Software *AlphaGo* von *Google DeepMind* in den Fokus der Wissenschaft und Öffentlichkeit. *AlphaGo* ist die erste Software, die Menschen im asiatischen Spiel *Go* schlagen konnte (Silver et al., 2016). Das Programm ist nicht explizit für *Go* programmiert, sondern lernte das Spiel komplett in der Simulation bzw. im Spiel gegen sich selbst. Das Konzept wurde mit *AlphaZero* auf *Shogi* und Schach erweitert, wobei in allen Fällen die gleiche Software eingesetzt wird, die dann in einer anderen Umgebung trainiert wird (Silver et al., 2017b).

Im Folgenden wird zuerst in Kapitel 3.2.1 die mathematische Grundlage für den Lernprozess gelegt. In Kapitel 3.2.2 werden unterschiedliche RL-Agenten vorgestellt und verglichen, die alle auf DL basieren. Daraus wird die Entscheidung für den DQN-Agenten abgeleitet, der dann in Kapitel 3.2.3 detailliert vorgestellt wird.

3.2.1. Markow-Entscheidungsprozess[1]

Ein Markow-Entscheidungsprozess ist eine sequenzielle Optimierung eines zeitlich diskretisierten, stochastischen Systems. Die Zustandsänderungen des Systems können über die Zeit beeinflusst werden. Das Ziel ist die Auswahl eines Steuerungsmechanismus, der eine Bewertungs- bzw. Belohnungsfunktion maximiert. Ein Markow-Entscheidungsprozess ist vollständig über Zustands- und Aktionsraum, Bewertungsfunktion, Übergangswahrscheinlichkeiten und Wahrscheinlichkeiten der Anfangszustände definiert. Anschließend werden diese im Rahmen des RL eingeführt.

Im RL interagiert ein lernender Agent mit einer Umgebung, die das zu optimierende System darstellt. Der Agent beobachtet die Umgebung, führt darin Aktionen aus und erhält für diese von der Umgebung Belohnungen bzw. Bewertungen. Die Interaktion von Agent und Umgebung erfolgt in diskreten Zeitschritten $t \in \mathbb{N}$. Innerhalb jedes Zeitschrittes erhält der Agent den Zustand, d.h. die Beobachtung $s_t \in \mathcal{S}_t$ der Umgebung. Aufgrund

[1] Das Kapitel basiert auf den Quellen Sutton et al. (1998, S. 51 ff.) sowie Kallenberg (2002).

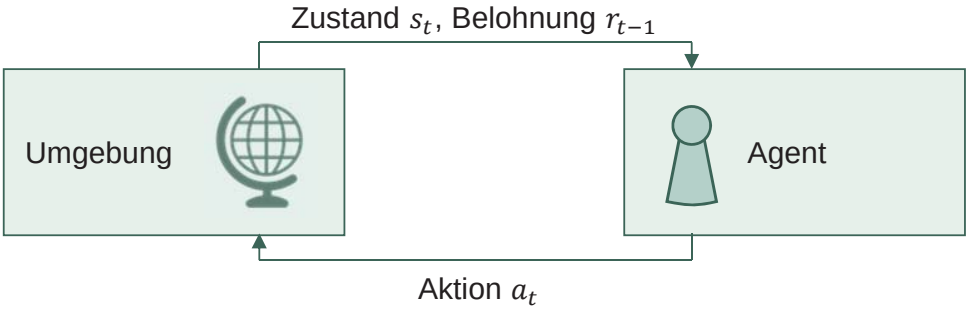


Abbildung 3.3.: Umgebung und Agent in einem MARKOW-Entscheidungsprozess

der Beobachtung wählt der Agent mithilfe seiner *Policy* $\pi(s_t)$ eine Aktion $a_t = \pi(s_t) \in \mathcal{A}_t$. Die Räume $\mathcal{S}_t$ und $\mathcal{A}_t$ bezeichnen dabei den Zustandsraum und Aktionsraum. Im nächsten Schritt bekommt der Agent zusätzlich zum Zustand $s_{(t+1)}$ die Belohnung $r_t = r(s_t, a_t) \in \mathcal{R} \subset \mathbb{R}$. Abbildung 3.3 gibt einen Überblick über die Interaktion zwischen Umgebung und Agent.

In einem finiten MARKOW-Entscheidungsprozess haben Zustands-, Aktions- und Belohnungsraum nur eine endliche Anzahl an Elementen. In diesem Fall existieren feste Übergangswahrscheinlichkeiten, die nur von dem vorherigen Zustand und der vorherigen Aktion abhängen:

$$p\left(s', r \;\middle|\; s, a\right) := \Pr\left\{s_t = s',\ r_{t-1} = r \;\middle|\; s_{t-1} = s,\ a_{t-1} = a\right\}. \tag{3.5}$$

Mit dieser Definition der Wahrscheinlichkeitsfunktion p gilt:

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p\left(s', r \;\middle|\; s, a\right) = 1, \quad \forall s \in \mathcal{S},\ a \in \mathcal{A}(s). \tag{3.6}$$

Das Tupel $\left(\mathcal{S}, \mathcal{A}, r(\cdot,\cdot), p(\cdot,\cdot), p_0\right)$ definiert den MARKOW-Entscheidungsprozess vollständig, wobei p_0 die Wahrscheinlichkeiten der verschiedenen Anfangszustände s_0 ist.

3.2.2. Vergleich unterschiedlicher RL-Agenten

Deep Reinforcement Learning (DRL) revolutionierte das RL und stellt einen wichtigen Schritt in Richtung autonomer Systeme dar (Arulkumaran et al., 2017). Als Forschungsfeld existierte RL bereits vor DL und konnte erste Erfolge vorweisen, beispielsweise den SARSA-Agenten (*State–Action–Reward–State–Action*; Rummery et al., 1994) oder durch *Q-learning* (Watkins et al., 1992). Allerdings brachte erst die Kombination mit DL den Durchbruch: Die auf DL basierenden Agenten skalierten zu realistischen, hochdimensionalen Problemgrößen, die vorher unerreichbar waren (Arulkumaran et al., 2017). Die volle Leistungsfähigkeit des Ansatzes wurde durch *AlphaGo* demonstriert, die erste Software, die einen Weltmeister im asiatischen Brettspiel *Go* schlagen konnte (Silver et al., 2016). Aufgrund der höheren Leistungsfähigkeit beschränkt sich die folgende Diskussion auf DL basierte RL-Agenten, die das Forschungsfeld des DRL bilden. Es werden insbesondere die DRL-Agenten besprochen, die in der Literaturübersicht von Arulkumaran et al. (2017) als zentrale oder Benchmark Algorithmen des Forschungsfelds angesehen werden.

Der Durchbruch von DRL wurde durch den DQN-Agenten eingeleitet (Arulkumaran et al., 2017; Mnih et al., 2013; Mnih et al., 2015). Der DQN-Agent erreichte bei einer Reihe von *Atari*-Spielen ein Level, das die Spielstärke menschlicher Spieler übertraf. Besonders war die Tatsache, dass er ausschließlich direkt auf dem Rohsignal, also den Pixeln des Videospiels, trainiert wurde. Diese Eigenschaft ist typisch für DL-Systeme, die direkt auf Rohdaten anstatt von entwickelten *Features* lernen (*representation learning*; LeCun et al., 2015).

Das direkte Trainieren auf Rohdaten erfordert meistens die Fähigkeit, mit kontinuierlichen Räumen arbeiten zu können. Alle DRL-Agenten können sowohl mit diskreten als auch kontinuierlichen Zustandsräumen umgehen. Klassische Ansätze wie SARSA oder tabellarisches RL sind auf diskrete Probleme limitiert. Der Zustandsraum des autonomen *Dispatchings* wird sowohl diskrete als auch kontinuierliche Werte beinhalten: Die verbleibende Zeit

bis zum Liefertermin ist ein kontinuierlicher Wert, während der Produkttyp eines Auftrags diskret festgelegt wird.

Neben dem Zustandsraum kann auch der Aktionsraum diskret oder kontinuierlich sein. Ein Beispiel für einen kontinuierlichen Zustandsraum ist das Steuern eines autonomen Fahrzeugs mit einem kontinuierlichen Lenkeinschlag. Da beim autonomen *Dispatching* ein konkreter Auftrag aus einer Auftragsmenge gewählt wird, ist der Aktionsraum diskret. Daher stellt es keine Einschränkung dar, dass die erste Implementierung des DQN-Agenten nur für diskrete Aktionsräume konzipiert war (Mnih et al., 2015). Die meisten DRL-Agenten, u.a. die Erweiterungen des DQN-Agenten (z.B. Gu et al., 2016), können jedoch mit kontinuierlichen und diskreten Aktionsräumen arbeiten (Arulkumaran et al., 2017).

Prinzipiell wird zwischen modellfreiem und modellbasiertem RL unterschieden (engl. *model-free/-based RL*). Modellfrei bedeutet, dass der RL-Agent kein explizites Modell seiner Umgebung mit Übergangswahrscheinlichkeiten erlernt. Ein modellbasierter RL-Agent erlernt dieses Modell und kann bei einem hinreichend guten Modell selbst Simulationen der Umgebung vornehmen. Theoretisch benötigt ein modellbasierter RL-Agent daher weniger Interaktionen mit der Umgebung. Gerade bei physischer Ausführung der Aktionen, beispielsweise in der Robotik, hat ein modellbasierter RL-Agent Vorteile. Bisher führten die theoretischen Vorteile der modellbasierten RL-Agenten jedoch zu keinem empirischen Vorsprung gegenüber modellfreien RL-Agenten (Arulkumaran et al., 2017). Das Gegenteil ist der Fall: Die einfacheren, modellfreien Ansätze können die Rechenzeit besser ausnützen und erzielen die besseren Resultate (Arulkumaran et al., 2017). Alle hier besprochen RL-Agenten sind modellfrei.

Das Lernen kann prinzipiell *On-Policy* und *Off-Policy* stattfinden. Diese Eigenschaft eines Algorithmus bezieht sich darauf, ob die ausgeführten Aktionen in der Umgebung mit der lernenden *Policy* oder einer anderen *Policy* ermittelt wurden (François-Lavet et al., 2018). Die *On-Policy*-Algorithmen testen und verbessern ihre Strategie direkt, während die *Off-Policy*-Agenten ihre erste *Policy* für die Generation der Daten von der zweiten, lernenden *Policy* trennen. Die *Off-Policy*-Methoden können somit meistens schneller

lernen, da sie aus jeder Form von Erfahrung in der Umgebung lernen können, also beispielsweise auch aus zufälligen Aktionen (François-Lavet et al., 2018). Aufgrund dieser Eigenschaft werden *Off-Policy*-Algorithmen in dieser Forschungsarbeit bevorzugt.

Die RL-Agenten nutzen unterschiedliche Wertefunktionen, um die Güte einer Aktion abzuschätzen. Dabei optimieren die meisten Agenten nicht die direkte Bewertung, sondern die langfristige Rückgabe R (engl. *return*):

$$R = \sum_{t=0}^{T-1} \gamma^t \cdot r_{t+1}, \tag{3.7}$$

bei einer Zeitspanne T bis zum Ende einer Episode, beispielsweise die Anzahl der Züge bis zum Ende eines Schachspiels. Der Faktor γ wird als *Discount*-Faktor bezeichnet, der sofortige und langfristige Bewertungen gegeneinander abwiegt. Die *State-Value*-Funktion $V^\pi(s)$ ergibt sich aus der erwarteten Rückgabe R unter einer festen Strategie π ausgehend vom Ausgangszustand s:

$$V^\pi(s) = \mathbb{E}[R \mid s, \pi]. \tag{3.8}$$

Die *State-Value*-Funktion kann damit auch für die Bewertung eines Zustands herangezogen werden, unter der Annahme, dass der Strategie π bis zum Ende gefolgt wird. Die Qualitätsfunktion, meistens als Q-Funktion abgekürzt, erweitert dieses Konzept. Sie bewertet Zustand s und jede Aktion a in diesem Zustand unter der Annahme, dass danach der *Policy* π gefolgt wird:

$$Q^\pi(s, a) = \mathbb{E}[R \mid s, a, \pi]. \tag{3.9}$$

Die Q-Funktion repräsentiert damit absolute Zustands-Aktions-Werte. Häufig ist es für das Lernen des Agenten jedoch einfacher, nur den Vorteil zu kennen, der durch eine bestimmte Aktion erzielt wird. Dabei sollte nicht nur die direkte Bewertung r_t mit einbezogen werden, sondern ebenfalls die Güte eines Zustands s (also z.B. wie aussichtsreich der Zustand für einen

Sieg wäre). Aus dieser Überlegung heraus wird die *Advantage*-Funktion formuliert:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s). \tag{3.10}$$

Die meisten RL-Agenten nutzen die *Q*-Funktion oder die *Advantage*-Funktion zur Beurteilung von Aktionen.

Diese Beurteilungen werden für das Lernen herangezogen. Ein RL-Agent erlernt eine (oder mehrere) der folgenden Komponenten (François-Lavet et al., 2018):

- eine Repräsentation einer *Value Function*, die Vorhersagen über die erwartete Bewertung bzw. Belohnung bei einer bestimmten Aktion macht.
- eine direkte Repräsentation einer *Policy* π, die für einen Zustand eine Aktion auswählt.
- ein Modell der Umgebung, in der ein Planungsalgorithmus die nächste Aktion bestimmt.

Die ersten beiden Ansätze beziehen sich auf das modellfreie RL, der letzte Ansatz auf das modellbasierte RL. Wie bereits diskutiert übertrifft die Leistungsfähigkeit von modellfreien Ansätzen aktuell die der modellbasierten, weshalb der Fokus auf den ersten beiden Ansätzen liegt. Für die Kategorisierung der Benchmark-Agenten spielt eine weitere Kategorie eine wichtige Rolle, die eine hybride Methode aus *Value-Function*- und *Policy-Search*-Ansätzen bildet: der *Actor-Critic*-Ansatz (Arulkumaran et al., 2017). Teilweise wird der *Actor-Critic*-Ansatz als Unterkategorie des *Policy-Search*-Ansatzes gesehen, da die Entscheidungen direkt mit der *Policy* bestimmt werden. Hier wird er einzeln aufgelistet, da er mehrere erfolgreiche Algorithmen hervorgebracht hat. Die drei erfolgreichsten Ansätze des DRL sind damit (Arulkumaran et al., 2017):

- ***Value-Function*-Ansatz.** Der RL-Agent erlernt ein Modell zur Vorhersage der Beurteilung. In den letzten Jahren wurde meistens die *Advantage*-Funktion anstatt der *Q*-Funktion genutzt (Arulkumaran et al., 2017). Der erste DRL-Ansatz, der DQN-Agent, fällt in diese Kategorie. Seit der Veröffentlichung des DQN-Agenten wurden diverse Erweiterungen vorgeschlagen, u.a. *double Q-learning* (Hasselt et al., 2016), *prioritized replay* (Schaul et al., 2016) und *dueling networks* (Wang et al., 2016). Die Erweiterungen und Verbesserungen wurden alle im *Rainbow*-Agenten zusammengeführt (Hessel et al., 2018). Der *Normalized-Advantage-Function*-Algorithmus basiert ebenfalls auf dem DQN-Agenten mit den Erweiterungen, arbeitet jedoch auch auf einem kontinuierlichen Aktionsraum Gu et al. (2016).
- ***Policy-Search*-Ansatz.** Wie bereits erwähnt, wird eine direkte Repräsentation der *Policy* erlernt. Ein populärer Ansatz ist der *Trust-Region-Policy-Optimization (TRPO)*-Algorithmus, bei dem die *Policy* gleichmäßig innerhalb einer *trust region* verändert bzw. erweitert wird (Schulman et al., 2015). Der *Generalized-Advantage-Estimation (GAE)*-Algorithmus, der auf TRPO basiert, reduziert weitere Varianzen im Lernprozess, wodurch er stabiler wird (Schulman et al., 2016). Eine kürzliche Entwicklung ist der *Proximal-Policy-Optimization (PPO)*-Algorithmus, der ebenfalls auf TRPO basiert. Der PPO-Algorithmus zeichnet sich bei gleichbleibender Leistung durch eine einfache Implementierung und wenige Hyperparameter aus (Schulman et al., 2017).
- ***Actor-Critic*-Ansatz.** Dieser RL-Ansatz nutzt sowohl eine Repräsentation der *Policy* (*actor*) als auch eine *Value*-Funktion. Der *Actor* lernt, indem er nicht direkt die Beurteilung r_t nutzt, sondern das Feedback seines *Critic*. Die *Actor-Critic*-Algorithmen profitieren sowohl von Weiterentwicklungen der *Value-Function*-Ansätze als auch der *Policy-Search*-Ansätze. Dafür sind sie in ihrer Implementierung und Einstellung deutlich komplizierter. Die zwei Benchmark-Methoden sind der *Deep-Deterministic-Policy-Gradient (DDPG)*-Agent (Lillicrap et al., 2016),

der auf dem *Deterministic-Policy-Gradient (DPG)*-Algorithmus (Silver et al., 2014) aufbaut, und der *Asynchronous-Advantage-Actor-Critic (A3C)*-Algorithmus (Mnih et al., 2016).

Tabelle 3.1 gibt einen Überblick über die bisher vorgestellten RL-Agenten. Die Algorithmen *Q-learning* und SARSA wurden der Vollständigkeit halber aufgenommen, obwohl sie keine DRL-Agenten sind. Im Gegensatz zu den DRL-Agenten unterstützen beide keinen kontinuierlichen Zustandsraum.

In diesem Forschungsprojekt wurde die Entscheidung für einen RL-Agenten im autonomen *Dispatching* im Jahr 2015 gefällt. Wegen der hohen Leistungsfähigkeit der DL-basierten Ansätze und des kontinuierlichen Zustandsraumes war schnell klar, dass ein DRL-Agent genutzt werden sollte. Damals war die Auswahl an DRL-Agenten allerdings beschränkt. Eine gute Dokumentation und verfügbare *Open-Source-Code*-Beispiele spielten bei der Entscheidung ebenfalls eine wichtige Rolle, um sich dem komplizierten Thema zu nähern. Schlussendlich fiel die Entscheidung auf den DQN-Agenten. Es zeichnete sich bereits ab, dass viele weitere Arbeiten und Verbesserungen auf diesem revolutionären DRL-Agenten basieren würden. Im jungen, dynamischen DL-Forschungsfeld hatte er bereits die Zeiten überdauert und stand als Meilenstein der DRL-Entwicklung fest. Er erfüllte die Anforderungen des Forschungsprojekts und das Setzen der Hyperparameter war verhältnismäßig gut dokumentiert.

Aus heutiger Sicht (Stand 2018) lässt sich sagen, dass inzwischen leistungsfähigere DRL-Agenten verfügbar sind. Trotzdem hat sich die Entscheidung für den DQN-Agenten als gute Wahl herausgestellt. Es wurden viele Erweiterungen veröffentlicht, die sich in den DQN-Agenten integrieren lassen.

Für weitere Forschungsarbeiten wäre es interessant, die Leistungsfähigkeit der neuen Methodik unter Nutzung unterschiedlicher RL-Agenten zu testen. Bei der Implementierung wurde auf das universelle *OpenAI GYM-Interface* geachtet, das einen Austausch des Agenten ermöglicht. Eine interessante Alternative zum DQN-Agenten wäre der PPO, da seine Implementierung und das Setzen der Hyperparameter relativ einfach sind. Dies könnte das

Algorithmus	Abkürzung	Jahr der Veröffentlichung	On-/Off-Policy	*Advantage (A)-/ Q-Funktion*
Q-learning	-	1992	off	Q
State–Action–Reward–State–Action	SARSA	1994	on	Q
Deep Q Network	DQN	2013	off	Q
Trust Region Policy Optimization	TRPO	2015	on	A
DQN with double Q-learning	-	2016	off	Q
DQN with prioritized replay	-	2016	off	Q
DQN with dueling networks	-	2016	off	Q
Normalized Advantage Functions	NAF	2016	off	A
Asynchronous Advantage Actor-Critic	A3C	2016	off	Q
Deep Deterministic Policy Gradient	DDPG	2016	off	Q
Generalized Advantage Estimation	GAE	2016	on	A
Proximal Policy Optimization	PPO	2017	on	A
Rainbow	-	2018	off	Q/A

DRL-Algorithmen klassische RL-Algorithmen

Tabelle 3.1.: Überblick über unterschiedliche RL-Agenten, die im Text vorgestellt wurden. Die RL-Agenten wurden von ARULKUMARAN et al. (2017) als zentrale Durchbrüche im Feld oder aktuelle Benchmark-Agenten benannt.

Hyperparameter-*Tuning* vereinfachen, was für Anwender der autonomen Produktionssteuerung ein Vorteil wäre.

3.2.3. Der *Deep-Q-Network*-Agent

Im Folgenden wird der DQN-Agent besprochen, der im autonomen *Dispatching* eingesetzt wurde. Der DQN-Agent erlernt eine Strategie zur Belohnungsmaximierung unter Verwendung von DL (Mnih et al., 2013; Mnih et al., 2015). Der DQN-Agent nutzt dabei *Q-learning* (Watkins et al., 1992), eine modellfreie Technik des RL, die zur Auswahl einer optimalen *Policy* π in einem Markow-Entscheidungsprozess genutzt wird. Die Q-Funktion ist eine Abschätzung der zukünftigen, gewichteten Bewertung des Systems bei einer festen *Policy* π.

Beim DQN-Agent wird eine Variante des *Q-learning* eingesetzt, die auch als *deep Q-learning* bezeichnet wird. Die Q-Funktion ist definiert als die zukünftig erwartete, kumulierte, gewichtete Belohnung:

$$Q(s,a) = \mathbb{E}\Big[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \Big| s_t = s, a_t = a, \pi \Big], \tag{3.11}$$

wobei $\gamma < 1$ den *Discount*-Faktor und π die Strategie bezeichnet, mit der die Aktionen ausgewählt werden. Es ist wichtig zu beachten, dass es sich bei den Belohnungen r um die zukünftigen Belohnungen handelt. Der DQN-Agent versucht, sich mit seiner Strategie immer in eine günstige Lage zu versetzen. Die Strategie π wird so gewählt, dass die Q-Funktion den optimalen Q-Wert Q^* approximiert:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\Big[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \Big| s_t = s, a_t = a, \pi \Big]. \tag{3.12}$$

Der DQN-Agent nutzt ein neuronales Netz, um für jede Aktion den zu erwartenden Q-Wert vorherzusagen. Die Aktion, die den besten Q-Wert verspricht, wird ausgewählt.

Das NN ist unabhängig vom RL-Algorithmus. Es muss in der Lage sein, die Q-Funktion abzubilden. In Mnih et al. (2015) besteht das NN aus drei *Convolutional-Layern* und einem abschließenden *Fully-Connected-Layer*. Eine vereinfachte Übersicht über den DQN-Agenten ist in Abbildung 3.4 gezeigt.

Zustände, Aktionen und Belohnungen werden gespeichert und immer wieder verwendet, um den Agenten weiter zu trainieren (engl. *experience*

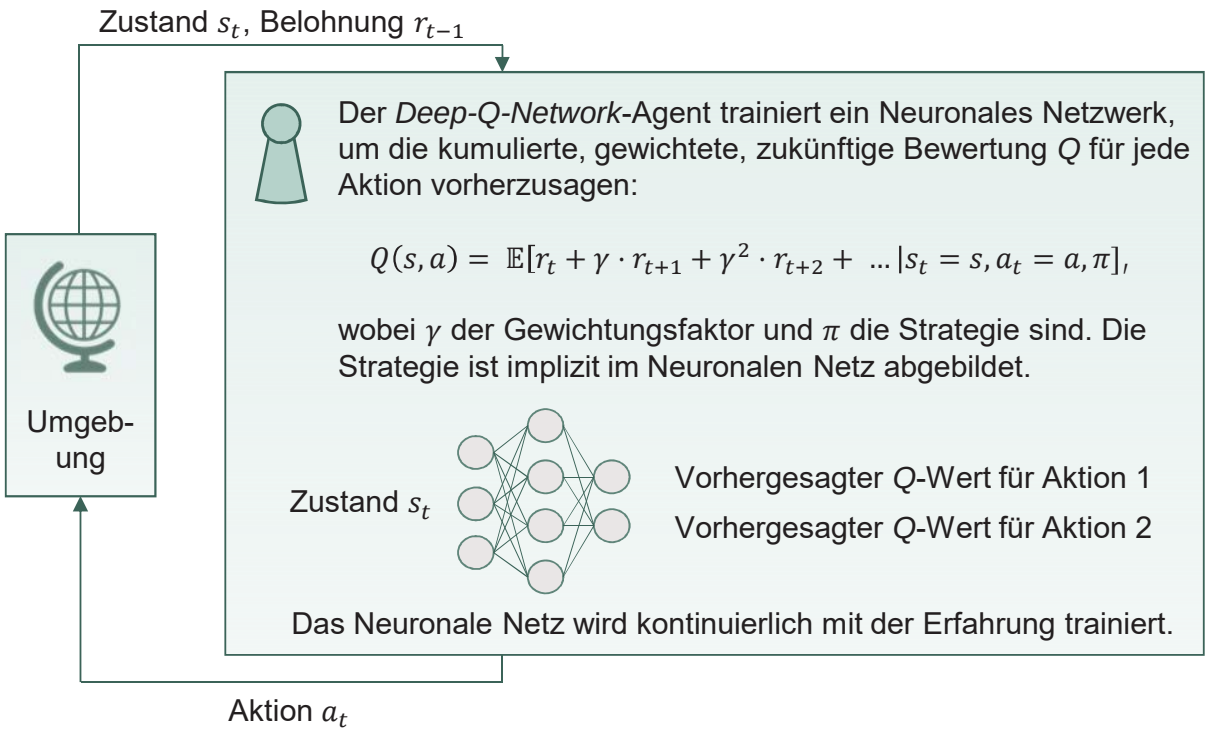


Abbildung 3.4.: Funktionsweise des DQN-Agenten

replay). In jedem Trainingsschritt wird aus dem Speicher $D_t = \{e_1, \ldots, e_t\}$ zufällig eine feste Anzahl an Erfahrungen $e_t = (s_t, a_t, r_t, s_{t+1})$, also ein Tupel aus Zustand, Aktion, Belohnung und folgendem Zustand, ausgewählt. Dabei handelt es sich nicht um eine zeitliche Sequenz zeitlich aufeinanderfolgender Erfahrungen. Zeitliche Sequenzen neigen häufig dazu, ähnliche Trainingsdaten zu enthalten, was zu einer Überanpassung (engl. *overfitting*) des NN führt. Die Speichergröße des DQN-Agenten ist begrenzt (siehe auch Tabelle 3.2, Hyperparameter: *replay memory size*). Ist die maximale Speichergröße erreicht, werden die ältesten Erfahrungen schrittweise gelöscht.

Eine weitere Besonderheit des DQN-Agenten ist die Verwendung eines zweiten, separaten Ziel-Netzwerkes, das zur Vorhersage der Q-Werte genutzt wird. Es ist identisch mit dem ersten Netzwerk und wird periodisch oder kontinuierlich teilweise durch dieses aktualisiert. Diese Verwendung führt zu einer Stabilisierung des Trainings, da der DQN-Agent einen festen Zeitraum mit einer gleichbleibenden oder zumindest ähnlichen Strategie arbeitet. Nur dadurch lässt sich langfristiges Handeln sicherstellen.

Prinzipiell basiert die Idee dafür auf der Bellman-Gleichung, die sich aus Gleichung 3.12 herleiten lässt:

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma \cdot \max_a \left(Q^*(s_{t+1}, a) \right) \mid s_t, a_t \right]. \tag{3.13}$$

Die Bellmann-Gleichung wird genutzt, um iterativ die Aktions-Bewertungs-Funktion anzunähern. Dafür wird ein iteratives Update definiert:

$$Q_{i+1}(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma \cdot \max_a \left(Q_i(s_{t+1}, a) \right) \mid s_t, a_t \right]. \tag{3.14}$$

Offensichtlich konvergiert $Q_i \to Q^*$ bei $i \to \infty$. Im Folgenden wird Q_{i+1} als Q und Q_i als $\hat{Q}$ bezeichnet. Während Q trainiert wird, wird $\hat{Q}$ konstant gehalten. Mit dem Parameter *target network update frequency* wird bestimmt, nach welcher Anzahl an Schritten $\hat{Q} = Q$ das Ziel-Netzwerk aktualisiert wird.

Der Trainingsalgorithmus für das NN des DQN-Agenten kann frei gewählt werden. In der ersten Publikation wurde *RMSProp* verwendet (Mnih et al., 2015). In dieser Arbeit wird der neuere *Adam*-Algorithmus eingesetzt, der in Experimenten zu einer schnelleren Konvergenz als *RMSProp* führte (Kingma et al., 2015). Auf die Hyperparameter, die spezifisch für *RMSProp* sind und in der ersten Publikation des DQN-Agenten verwendet wurden, wird hier daher nicht näher eingegangen (siehe Tabelle 3.2, Hyperparameter: *learning rate, gradient momentum, squared gradient momentum, min squared gradient*; für Details siehe Tieleman et al. (2012)).

Der Hyperparameter *minibatch size* in Tabelle 3.2 bezieht sich auf die Anzahl der Trainingsbeispiele, nach denen die Gewichte angepasst werden. Der Parameter ist ein Standard beim Training von NNs, um die Trainingsgeschwindigkeit zu erhöhen.

Der DQN-Agent nutzt eine sogenannte *ϵ-greedy policy*, um seine Umgebung zu erkunden. Mit einer Wahrscheinlichkeit ϵ wird die Aktion nicht vom NN bestimmt, sondern zufällig ausgewählt. Mithilfe der zufälligen Aktionen kann der DQN-Agent neue Strategien entdecken. Normalerweise wird der Hyperparameter ϵ während des Trainings linear mit der Anzahl der Trainingsschritte abgesenkt (*linear annealed ϵ-greedy policy*). Da das NN anfangs

kein Wissen über die Umgebung hat, wird üblicherweise mit $\epsilon = 1$, dem Hyperparameter *initial exploration*, gestartet. Der ϵ-Parameter sollte nie auf 0 abgesenkt werden, da sonst eine Überanpassung droht. Daher wird ein minimales ϵ bestimmt, üblicherweise $\epsilon_{\text{min}} = 0,1$, das nicht unterschritten wird. Im DQN-Algorithmus wird dieser Hyperparameter ϵ_{min} mit *final exploration* bezeichnet. Er wird nach der Schrittanzahl, die mit dem Hyperparameter *final exploration frame* festgelegt wird, erreicht.

Der DQN-Agent wurde für Atari-Videospiele entwickelt. Für eine Steuerung des Videospiels ist es nicht entscheidend, jeden *frame* einzeln wahrzunehmen. Mit dem Hyperparameter *action repeat* wird angegeben, dass der DQN-Agent seine Aktion nur bei jedem x-ten *frame* neu bestimmt. Mit der *agent history length* wird bestimmt, wie viele der letzten *frames* der DQN-Agent für seine Entscheidungsfindung erhält. Für die Atari-Spiele wurde eine Sequenz der in *agent history length* bestimmten Länge als ein Zustand definiert, da aus einem *frame* beispielsweise nicht immer die Bewegungsrichtung eindeutig abzulesen ist.

Am Anfang jeder Episode (bzw. jedes Spiels) wird eine bestimmte Anzahl an zufälligen Aktionen ausgeführt. Dies dient dazu, den DQN-Agenten jeweils in einer neuen Spielsituation starten zu lassen. Die Anzahl der Schritte wird im Hyperparameter *replay start size*, Tabelle 3.3, festgelegt. Damit der DQN-Agent nicht zur Untätigkeit tendiert, aus der er wenig lernen kann, wird mit dem Hyperparameter *no-op max* festgelegt, wie lange der DQN-Agent anfangs keine Aktion ausführen darf.

Tabelle 3.2 und 3.3 geben einen Überblick über alle Hyperparameter des DQN-Agenten sowie die typische Größenordnung, in der die Parameter gewählt werden. Der vereinfachte Ablauf ist in Algorithmus 3.1 gezeigt.

Hyper-parameter	Wert	Beschreibung
minibatch size	32	Anzahl der Trainingsbeispiele, über die jedes Gradientenupdate berechnet wird
replay memory size	10^6	Die Trainingsbeispiele werden aus einem Erfahrungsspeicher dieser Größe zufällig ausgewählt.
agent history length	4	Länge der Sequenz der letzten Zustände, die das *Q*-Netzwerk erhält
target network update frequency	10 000	Frequenz (gemessen in der Anzahl an Parameter-Updates), mit der das Zielnetzwerk aktualisiert wird. Anmerkung: Bei Werten unter 1 wird das Zielnetzwerk kontinuierlich als Überlagerung der Netzwerke im Verhältnis des Parameters aktualisiert.
discount factor	0,99	Der *Discount*-Faktor γ wird zur Berechnung der *Q*-Funktion genutzt.
action repeat	4	Anzahl der Wiederholungen, mit der jede Aktion des DQN-Agenten ausgeführt wird. Ein Wert von 4 bedeutet, dass der Agent nur jeden 4ten Schritt sieht.
update frequency	4	Anzahl der Aktionen des DQN-Agenten zwischen Gradientenupdates des NN
learning rate	0,00025	Lernrate, die vom Optimierer *RMSProp* verwendet wird
gradient momentum	0,95	Impuls, der von *RMSProp* verwendet wird
squared gradient momentum	0,95	quadrierter Gradienten-Impuls, der von *RMSProp* verwendet wird
min squared gradient	0,01	Konstante, die zum quadrierten Gradienten im Nenner des *RMSProp* Algorithmus addiert wird
initial exploration	1	initialer Wert $\epsilon_{\max}$ in der ϵ-*greedy* Exploration
final exploration	0,1	finaler Wert $\epsilon_{\min}$ in der ϵ-*greedy* Exploration
final exploration frame	10^6	Anzahl der Schritte, in denen ϵ von $\epsilon_{\max}$ zu $\epsilon_{\min}$ linear abgesenkt wird

Tabelle 3.2.: Erster Teil der Hyperparameter und der verwendeten Werte in der ersten Publikation des DQN-Agenten. Basierend auf Mnih et al. (2015)

Hyper-parameter	Wert	Beschreibung
replay start size	50000	Anzahl der Schritte am Anfang, für die Aktionen zufällig ausgewählt werden, bevor das Lernen beginnt und die Erfahrungen gespeichert werden
no-op max	30	maximale Anzahl der „do nothing"-Aktion des DQN-Agenten am Anfang jeder Episode

Tabelle 3.3.: Zweiter Teil der Hyperparameter und der verwendeten Werte in der ersten Publikation des DQN-Agenten. Basierend auf Mnih et al. (2015)

Algorithmus 3.1 Simplified Deep Q-Learning with experience replay (Mnih et al., 2015).

Initialize replay memory D to capacity *replay memory size*
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For *episode*$= 1, M$ **do**
    Initialize state s_1
    For $t = 1, T$ **do**
        With probability ϵ select a random action a_t,
        otherwise select $a_t = \max_a Q(s_t, a; \theta)$
        Execute action a_t in simulator and observe reward r_t
        and state s_{t+1}
        Store experience $e_t = (s_t, a_t, r_t, s_{t+1})$ in D
        Sample random minibatch of experiences e_j from D
        Set $y_j = \begin{cases} r_j & \text{if episode terminates} \\ & \text{at step } j+1 \\ r_j + \gamma \max_a \hat{Q}(s_{t+1}, a; \theta^-) & \text{otherwise} \end{cases}$
        Perform a gradient descent step on $(y - Q(s_t, a_t; \theta))^2$ with respect to the network parameters θ
        Reset $\hat{Q} = Q$ with the *target network update frequency*
    End For
End For

3.3. Charakteristiken von Multi-Agenten-Systemen und -Umgebungen[1]

In dieser Arbeit werden mehrere DQN-Agenten zu einem MAS kombiniert. Daher wird eine kurze Charakterisierung von MAS und den entsprechenden Umgebungen gegeben.

Die Umgebung der Agenten lässt sich prinzipiell anhand folgender Eigenschaften beschreiben:

1. **Deterministisch vs. stochastisch:** In einer deterministischen Umgebung hat eine Aktion bei gleichem Ausgangszustand immer die gleiche Auswirkung. In einer stochastischen Umgebung führt sie mit gewissen Wahrscheinlichkeiten zu unterschiedlichen Zuständen.
2. **Statisch vs. dynamisch:** Bezieht sich auf die Kontrolle der Umgebung durch den Agenten. In einer statischen Umgebung werden alle Veränderungen durch Aktionen des Agenten ausgelöst, während in einer dynamischen Umgebungen Veränderungsprozesse außerhalb des Einflusses der Agenten ablaufen.
3. **Diskret vs. kontinuierlich:** Bezieht sich auf die Art der Aktionen der Agenten. Diskrete Aktionen lassen sich klar voneinander abgrenzen, während kontinuierliche Aktionen fließend ineinander übergehen.
4. **Episodisch vs. sequenziell:** In einer episodischen Umgebung führt ein Agent eine Reihe voneinander unabhängiger Entscheidungen aus. In einer sequenziellen Umgebung beeinflusst eine Entscheidung auch zukünftige Bewertungen und Entscheidungen des Agenten.

Die Reichweite der Wahrnehmung der Umgebung kann ebenfalls der Umgebung, allerdings auch dem Agenten zugeordnet werden. Es wird zwischen einer partiellen bzw. lokalen Wahrnehmung und einer vollständigen bzw. globalen Wahrnehmung der Umgebung unterschieden.

[1] Das Kapitel basiert auf der Standardliteratur von Russell et al. (2010, S. 41 ff. u. 666 ff.) sowie von Shoham et al. (2009, S. 22 ff. u. 273 ff.).

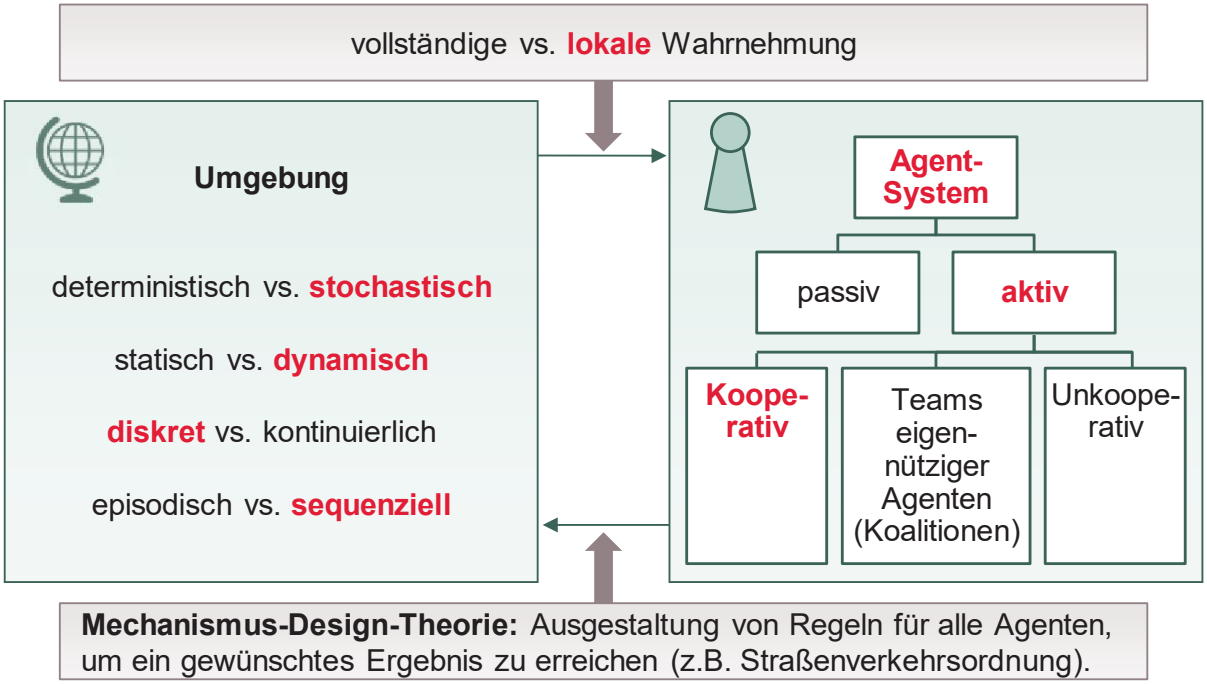


Abbildung 3.5.: Charakterisierung von MAS und der Umgebung. Rot markiert sind die Charakteristiken der verwendeten Umgebung in dieser Arbeit.

Ein Überblick über die Umgebungen ist in Abbildung 3.5 auf der linken Seite gegeben. Rot markiert sind die Eigenschaften der Umgebung in dieser Arbeit.

In MAS lassen sich Agenten in passiv und aktiv unterteilen. Passive Agenten untersuchen ihre Umgebung, ohne diese zu verändern oder zu beeinflussen. Die meisten Agenten sind aktiv und verändern durch ihre Aktionen die Umgebung. Die unterschiedlichen Agenten können bei der Verfolgung ihres Ziels unterschiedlich stark kooperieren. Prinzipiell wird zwischen kooperativen und unkooperativen bzw. kompetitiven Agenten unterschieden. Dazwischen gibt es den Fall, dass sich Agenten in Koalitionen zusammenschließen und sich in der Koalition unterstützen.

Die Mechanismus-Design-Theorie (MDT), ein Teilgebiet der Spieltheorie, beschäftigt sich mit der Ausgestaltung von Regeln für die Agenten. Sie legt fest, in welchem Rahmen bzw. welche diskreten Aktionen die Agenten ausführen können und welche Bewertung dafür gegeben wird. Mit der

MDT werden damit auch die Interaktionen zwischen den Spielern bzw. Agenten festgelegt. Bewertungen und der Raum der Aktionen müssen so gewählt werden, dass durch entsprechende Anreize und verfügbare Handlungen das gewünschte Gesamtergebnis erreicht werden kann.

Im Zusammenhang mit KI und ML wird anstatt von MDT meistens von *Reward Engineering* gesprochen (Dewey, 2014), wobei sich dies nur auf die Bewertung bezieht. Mit steigender Komplexität der Umgebung kommt der Gestaltung der Bewertungsfunktion eine immer wichtigere Rolle zu. Gleichzeitig wird die Formulierung der Bewertungsfunktion ebenfalls immer anspruchsvoller. Bewertungsfunktionen müssen teilweise auch ethische und moralische Argumente abbilden und gewichten. Der Informatiker und Philosoph Daniel Dewey bezeichnet diesen Zusammenhang als *Reward Engineering Principle* (Dewey, 2014).

Abbildung 3.5 gibt einen Überblick über Charakteristiken von Umgebungen und MAS. Außerdem wird dort eine Einordnung des MAS in dieser Arbeit gegeben. Eine komplexe Werkstattfertigung ist ein stochastisches, dynamisches System. Die Reihenfolgeabarbeitung ist diskret und sequenziell. Die Agenten, die die Produktionssteuerung übernehmen, handeln aktiv und beeinflussen das System durch ihre Aktionen. Die Agenten handeln kooperativ und verfolgen gemeinsam ein einheitliches Ziel.

Kapitel 4

Stand der Technik in der Halbleiterfertigung[1]

In diesem Kapitel wird der Stand der Technik anhand eines Beispiels aus der Halbleiterindustrie dargelegt. Dabei erhält der Leser einen Überblick über den Halbleiterfertigungsprozess in einer komplexen Werkstattfertigung.

Als konkretes Beispiel wird die Fertigungssteuerung der großen *Frontend*-Fertigungen[2] der Infineon Technologies AG beschrieben. Die Infineon Technologies AG ist der zweitgrößte europäischen Halbleiterhersteller nach Umsatz hinter NXP Semiconductors N.V. (IHS Markit, 2017). Das Unternehmen hat sich auf Leistungshalbleiter, Sensorik und Sicherheitsanwendungen spezialisiert. Den hohen Stand der Technik an diesen Fertigungsstandorten zeigt u.a. die Auszeichnung der Fertigung in Dresden in

[1] Das Kapitel basiert auf Expertengesprächen und der eigenen Tätigkeit im Unternehmen Infineon Technologies AG.

[2] Die Standorte sind Dresden und Regensburg (Deutschland), Villach (Österreich) und Kulim (Malaysia). Die Standorte machen über > 90 % des Eigenfertigungsvolumens der Infineon Technologies AG aus.

der Kategorie *„Hervorragende Standortsicherung durch Digitalisierung“* im Wettbewerb *„Fabrik des Jahres 2017“* (Hoffmann, 2017).

Zuerst wird in einem kurzen Exkurs in Kapitel 4.1 der Herstellungsprozess von Halbleitern erläutert, um dem nachfolgenden Beispiel und der Validierung in Kapitel 8 einen Kontext zu geben. Der Fokus dieses Teils liegt darauf, die notwendigen Arbeitsschritte für einen Chip darzulegen. Danach wird in Kapitel 4.2 die Organisation und Fertigungssteuerung einer modernen Scheibenfertigung anhand des Beispiels vorgestellt. Das Kapitel konzentriert sich auf die Ausführung der Arbeitsschritte.

4.1. Überblick über den Halbleiterfertigungsprozess

Elektronische Schaltungen auf einem kleinen (1-800mm^2), flachen Halbleiterkristall werden als *Integrated Circuits (ICs)* oder Chips bezeichnet. Im Herstellungsprozess werden sie durch das Aufbringen mehrerer Ebenen auf einer Halbleiterscheibe, einem Wafer, mit einem Durchmesser von bis zu 300 mm aufgebaut. Die Ebenen entstehen durch eine feste Abfolge von mehreren hundert lithographischen, mechanischen, elektrischen und chemischen Prozessschritten, die eine Produktionszeit von 11-13 Wochen erfordert. Dadurch werden elektrische Bauelemente, größtenteils Transistoren, auf dem Wafer prozessiert und miteinander verschalten. Für das Endprodukt wird der Wafer in einzelne Chips zersägt, die dann in Gehäusen eingefasst werden. Abbildung 4.1 gibt einen Überblick über die Größenordnungen.

Der erste Teil der Produktion, die Scheibenfertigung bzw. Waferprozessierung, erfolgt im *Frontend*, manchmal als FE abgekürzt, indem alle Schritte auf einem Wafer ausgeführt werden. Nach der Scheibenfertigung werden die Wafer im Scheibenprüffeld getestet und verlassen dann das *Frontend*. Die einzelnen elektrischen Bauteile auf dem Wafer werden jetzt als *Dies* bezeichnet. Die sogenannte *Die Bank* ist das Zwischenlager zwischen dem *Frontend* und dem *Backend*, abgekürzt als BE, in dem die Montage stattfindet. Der Ablauf bis zur *Die Bank* ist in Abbildung 4.2 gezeigt.

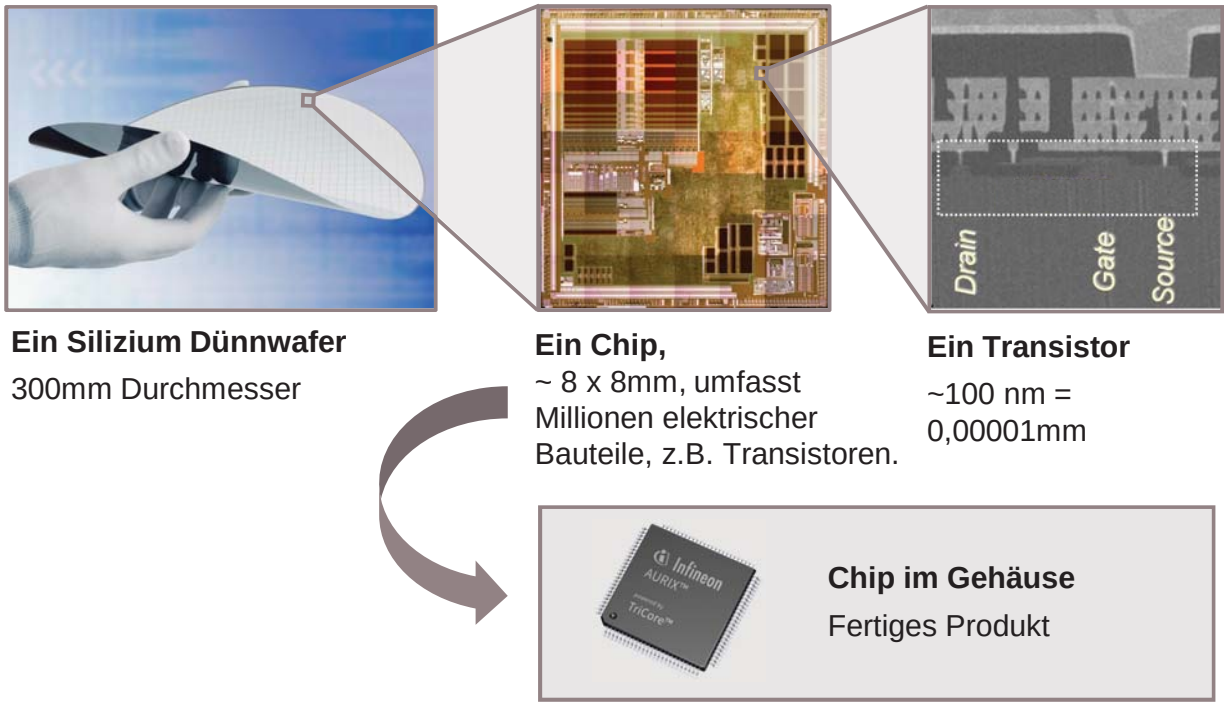


Abbildung 4.1.: Größenordnungen vom Wafer zum einzelnen Transistor. Quelle der Einzelbilder: Infineon Technologies AG

Im *Backend* wird der Wafer zersägt und in ein schützendes *Package* eingefügt. Das Package sorgt dafür, dass der Chip robust ist und weiterverarbeitet werden kann. Nach dem *Packaging* kann der Chip kontaktiert werden und im finalen Test auf seine Funktion getestet werden. Der *Backend*-Prozess ist in Abbildung 4.3 dargestellt.

Aufgrund der langen Durchlaufzeiten wird das *Frontend* häufig als *Make-to-Stock*-Fertigung beplant. Gehen Kundenaufträge ein, werden diese von der *Die Bank* im *Backend* gestartet. Das *Backend* arbeitet damit in einem *Make-to-Order*-Modus. Durch die *Die Bank* sind *Frontend* und *Backend* planerisch und logistisch entkoppelt. Dies erlaubt unterschiedliche Prioritätensetzungen im Management der Fabriken: Im kapitalintensiven *Frontend* wird stärker auf Auslastung und Stabilität geachtet, wogegen im *Backend* der Fokus auf Flexibilität liegt.

Diese Arbeit bezieht sich auf die Scheibenfertigung, die ausschließlich im Reinraum stattfindet. Daher wird auf das *Frontend* im Folgenden detaillierter eingegangen.

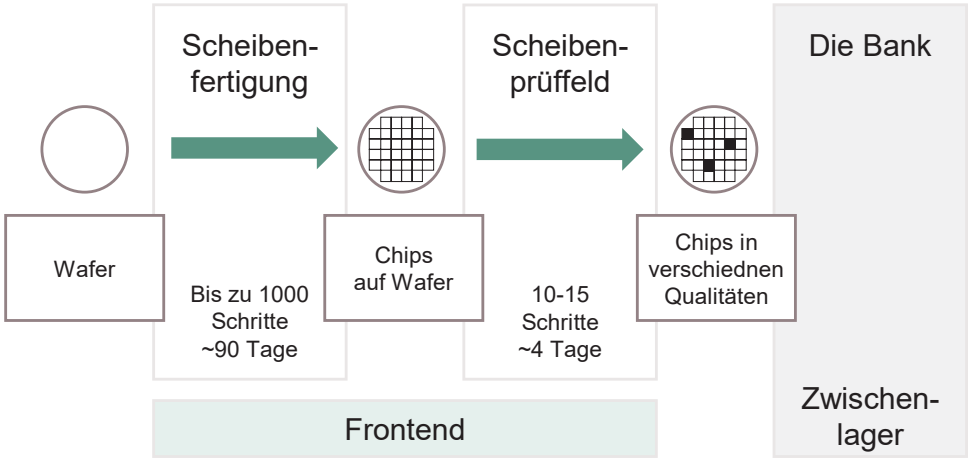


Abbildung 4.2.: Erster Teil der Lieferkette der Halbleiterfertigung (*Frontend*). Basierend auf Material der Infineon Technologies AG

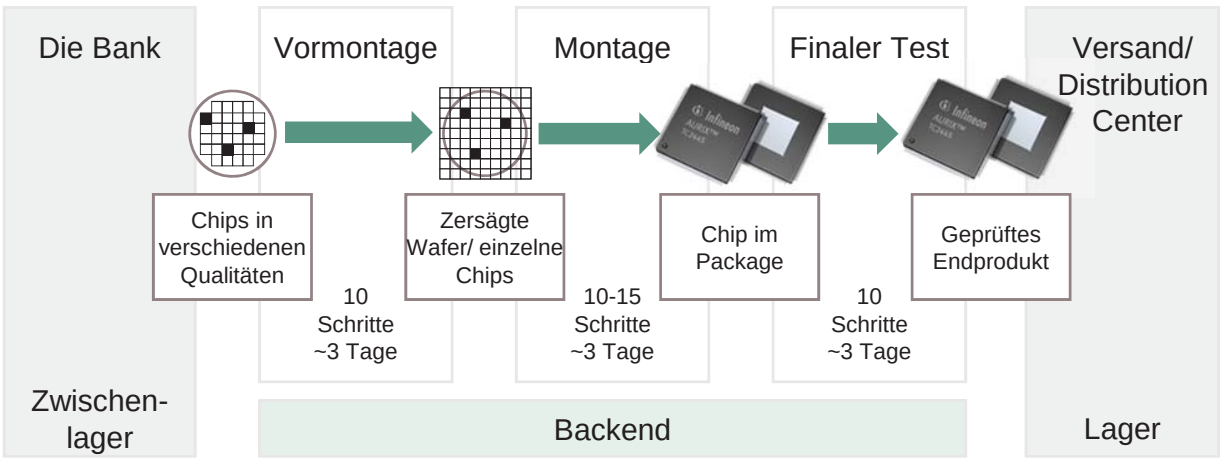


Abbildung 4.3.: Zweiter Teil der Lieferkette der Halbleiterfertigung (*Backend*). Basierend auf Material der Infineon Technologies AG

Auf das *Frontend* entfällt der Großteil der Arbeitsschritte und der DZ. Entscheidend für den Fokus auf das *Frontend* ist jedoch die Kapitalintensivität der Produktion. Im Jahr 2011 ermittelte McKinsey & Company Baukosten von US$3– US$4 Milliarden für eine 300 mm Halbleiterfabrik (Heck et al., 2011). Samsung kündigte 2015 den Bau einer neuen Fabrik an, in die über US$14 Milliarden investiert werden sollten (Johnson, 2015). Morris Chang,

Gründer der Taiwan Semiconductor Manufacturing Company (TSMC), rechnet damit, dass Halbleiter-*Frontends* zukünftig über US$20 Milliarden kosten werden (Burton, 2017).

Die Schritte in der Scheibenfertigung lassen sich in *Front-end-of-line* und *Back-end-of-line* unterteilen. Im *Front-end-of-line* werden die Transistoren geformt. Im *Back-end-of-line* erfolgt die logische Verschaltung durch die Metallisierungsebenen.

Der *Frontend*-Prozess besteht aus vier grundsätzlichen Prozessarten, die sowohl im *Front-end-of-line* als auch im *Back-end-of-line* angewandt werden: Deposition, Entfernen von Material, Strukturierung und Modifikation der elektrischen Eigenschaften. Im Folgenden werden die Prozessarten kurz besprochen.

- **Deposition.** Bei der Deposition wird eine neue Schicht an Material auf dem Wafer aufgebracht. Je nach Material der Schicht werden die Verfahren Rotationsbeschichtung, chemische oder physikalische Gasphasenabscheidung, elektrochemische Abscheidung, Molekularstrahlepitaxie oder Atomlagenabscheidung angewendet.
- **Strukturierung.** Bei der Fotolithographie wird das Bild einer Fotomaske mittels Belichtung auf einen Fotolack auf dem Wafer übertragen. Der Fotolack wird danach entwickelt, wobei je nach Fotolack die belichteten oder nicht-belichteten Bereiche des Fotolacks entfernt werden. Die Schritte belacken, belichten und entwickeln sind in der Fertigung in *Cluster-Tools* zusammengefasst. Durch die Strukturierung der Oberfläche kann nun lokal Material aufgebracht oder entfernt werden oder elektrische Strukturen lokal verändert werden. Der Fotolack wird danach wieder entfernt.
- **Entfernen von Material.** Häufig wird eine neue Materialschicht aufgebracht und dann mit einer darüber liegenden Schicht Fotolack strukturiert. Stellen, die nicht vom Fotolack bedeckt sind, werden dann aus der aufgebrachten Schicht entfernt. Hierfür werden Ätzprozesse eingesetzt. Wird zuerst strukturiert und dann Material abgeschieden, kann durch chemisch-mechanisches Polieren das Material auf dem

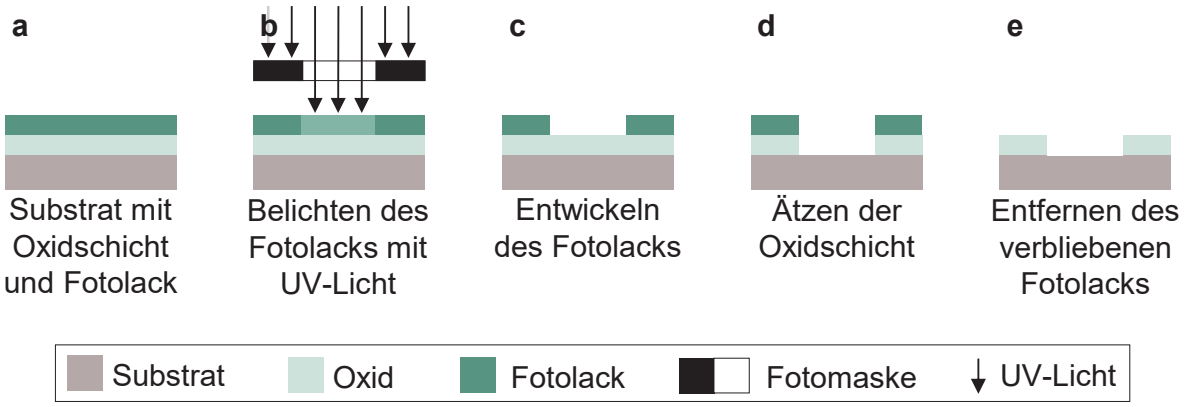


Abbildung 4.4.: Strukturierungsprozess des *Gate*-Oxids. In **a** werden Oxid und Fotolack aufgebracht. Danach wird in **b** der Fotolack UV-Licht ausgesetzt, der durch eine Maske nur bestimmte Bereiche erreicht. In **c** wird der Fotolack entwickelt, d.h. die belichteten Bereiche werden entfernt. Daraufhin wird in **d** die isolierende Oxidschicht an freigelegten Stellen weggeätzt. Somit werden gezielt isolierende Bereiche auf dem Wafer geschaffen. In **e** wird der verbleibende Fotolack durch ein Ätzverfahren entfernt, welches nur den Fotolack betrifft. Fotolack wird für jede strukturierende Schicht benötigt, ist jedoch nie Teil des finalen Prozessergebnisses.

Fotolack abgetragen werden, und die Bereiche zwischen dem Fotolack bleiben erhalten (*Damascene*-Prozess).

- **Modifikation der elektrischen Eigenschaften.** Dabei handelt es sich größtenteils um Ionenimplantation, einem Verfahren zum Einbringen von Fremdatomen, und die Oxidation. Der Implantation folgt immer ein Schritt in einem Ofen, bei dem die Fremdatome „eingebacken“ werden. Oxidation dient dem Bilden von Isolationsschichten, beispielsweise für das *Gate* im Transistor.

In Abbildung 4.4 ist die Strukturierung des *Gate*-Oxids gezeigt. Dabei werden die oben genannten Schritte angewendet, um zwischen Leitungskanal des Transistors und dem *Gate* eine isolierende Schicht aufzubringen.

4.2. Beispiel: Stand der Technik in der Halbleiterscheibenfertigung der Infineon Technologies AG in Dresden

Die Waferprozessierung in der Halbleiterfertigung stellt ein typisches Beispiel für eine komplexe Werkstattfertigung dar. Im Folgenden wird die Steuerung einer solchen Fabrik anhand des Beispiels der Infineon Technologies AG beschrieben. Wird der Begriff Halbleiterfertigung verwendet, bezieht es sich auf den kapitalintensiven Fertigungsanteil im *Frontend*, der nur im Reinraum möglich ist. Der einfachere Fertigungsanteil im *Backend*, das *Packaging* der Chips, wird nicht betrachtet.

Reinraumfläche ist im Bau und im Betrieb kostenintensiv, weshalb eine gute Flächenausnutzung erwünscht ist. In der Halbleiterfertigung werden zwei unterschiedliche Layouts für die Anordnung der Produktionsanlagen genutzt (Keil, 2012, S. 57): Das *Bay-Chase*-Layout (Puffer, 2007, S. 37) und das *Ballroom*-Layout (Chang et al., 1998; Weiss, 1999). Beim Bay-Chase-Layout sind die Produktionsanlagen entlang von Reinraum-Gängen angeordnet, von denen aus sie von Werkern oder Robotern beladen werden. Die Maschine selbst steht jedoch in einem Grauraum, der eine niedrigere Reinraumklasse benötigt. In diesem Raum können Instandhalter die Maschinen warten. Beim *Ballroom*-Layout stehen Maschinen frei im Raum und Werker und Instandhalter arbeiten auf der gleichen Fläche. Das Layout bietet eine höhere Flexibilität und bessere Kommunikationsmöglichkeiten. Allerdings ist die Kontaminationsgefahr bei Gasen und Partikelproblemen im *Ballroom*-Layout äußert hoch. Beide Layouttypen sind in Abbildung 4.5 gegenübergestellt.

Träger-Boxen für 200 mm Wafer sind offen bzw. nicht isoliert. Dagegen werden bei 300 mm Waferfertigungen fast ausschließlich isolierte *Front Opening Unified Pods (FOUPs)* (Doering et al., 2008, S. 33-3) verwendet, die an die Maschine angedockt und dann direkt geöffnet werden. Dadurch werden die Wafer nie direkt der Reinraumluft ausgesetzt.

Aufgrund der offenen oder nicht isolierten Boxen wird bei 200 mm Fertigungen meistens auf das bewährte *Bay-Chase*-Layout gesetzt. Dagegen wird bei 300 mm Fertigungen das flexiblere *Ballroom*-Layout angewendet. Die Infineon Technologies AG besitzt 200 mm und 300 mm Fertigungen und

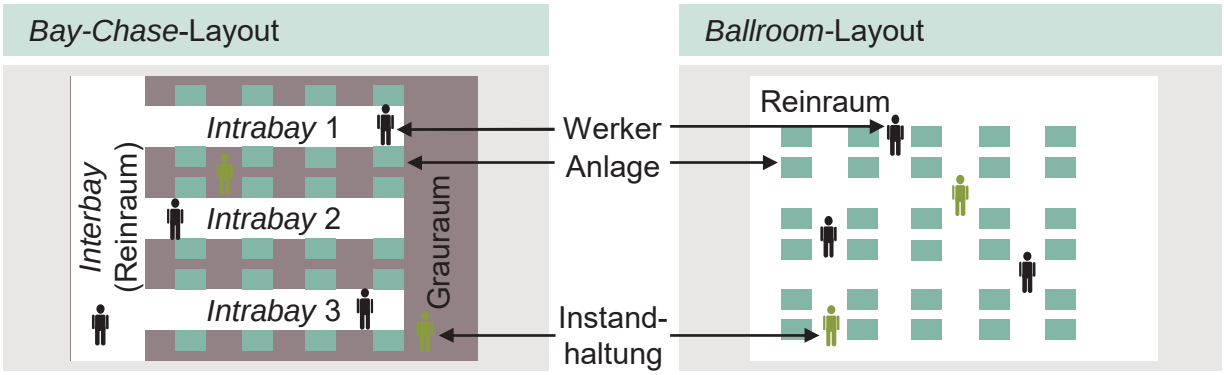


Abbildung 4.5.: Gegenüberstellung von *Bay-Chase*- und *Ballroom*-Layout. Basierend auf Keil (2012, S. 57)

verwendet beide Layouts. In Abbildung 4.6 ist der Blick in einen *Intrabay* der 200 mm Linie des Infineon Standorts in Dresden gezeigt.

In beiden Layouts werden Maschinen meistens nach dem Verrichtungsprinzip angeordnet, d.h. gleiche Anlagentypen sind räumlich konzentriert. Dies ist bemerkenswert, da dieser Organisationstyp bei kundenspezifischer Serienproduktion oder Massenproduktion untypisch ist (Wiendahl, 2014, S. 40). In der Halbleiterfertigung überwiegen jedoch die Vorteile für die Versorgung mit Fertigungsressourcen wie Gasen und die räumliche Konzentration der Linienexperten. In *More-than-Moore*-Fertigungen macht außerdem die Produktvielfalt mit unterschiedlichen Arbeitsplänen eine Organisation entlang von festen Arbeitsabläufen zunichte.

Die Automatisierungskonzepte für 200 mm Wafer und 300 mm Wafer Produktionen, im Halbleiterjargon als *Automated Material Handling System (AMHS)* bezeichnet, unterscheiden sich stark. Die 200 mm Fertigungen sind primär für den Betrieb durch Werker ausgelegt und wurden erst schrittweise nachträglich automatisiert. In den Fertigungen existiert ein zentrales Transportsystem mit Bahnhöfen an den *Bays*. Dort werden die Lose von Werkern oder Schienenrobotern auf die Anlagen im *Bay* verteilt. In Abbildung 4.7 (links) ist ein Schienenroboter gezeigt, der eine Seite eines *Bays* mit Aufträgen beliefert.

Wie die meisten 300 mm Fabriken wurde auch die Fertigung der Infineon Technologies AG in Dresden von Anfang an als hochautomatisierte

Abbildung 4.6.: Blick in einen *Intrabay* – häufig als „Finger“ bezeichnet – in der Reinraumproduktion. In den schwarzen Boxen werden jeweils 25 Wafer transportiert. Quelle: Infineon Technologies AG

Produktion geplant. Ein *Overhead Hoist Transfer (OHT)* System liefert die Lose direkt auf die Anlage aus und startet sie dort. Das *Overhead*-System funktioniert mit einzelnen Transporteinheiten, die mit Aufzügen ausgestattet sind, um Lose auf Maschinen abzustellen oder von dort aufzunehmen. In Abbildung 4.7 (rechts) ist ein OHT System in der 300 mm Linie der Infineon Technologies AG gezeigt.

Im Folgenden wird die Fertigungssteuerung besprochen. Es wird nicht auf die Planung eingegangen, die vor dem Start eines Auftrags in der Produktion stattfindet (siehe Kapitel 2.3). Sobald der Auftrag gestartet ist, übernehmen *Scheduling* und *Dispatching* die Reihenfolgebildung an den Maschinengruppen.

Abbildung 4.7.: Automatisierung in der 200 mm Linie (links) und der 300 mm Linie (rechts) in der Waferprozessierung. Im Bild links ist die charakteristische orangene Beleuchtung zu sehen, die in den Lithographie-Bereichen wegen des Fotolacks zum Einsatz kommt. Quelle: Infineon Technologies AG

Kern der Fertigungs-IT bildet das *Manufacturing Execution System (MES)*. Alle Maschinen, Roboter und das AMHS sind bei der Infineon Technologies AG an das MES angeschlossen. Im MES sind Ereignisse hinterlegt, die eine neue Reihenfolgeentscheidung notwendig machen. Ein Beispiel für ein Ereignis ist die Fertigstellung eines Prozessschritts, einer EPA , auf einer Maschine. Danach wird aus dem vorhandenen Umlaufbestand ein neuer Auftrag ausgewählt.

Beim Eintreten eines solchen Ereignisses ruft das MES den Applied Materials, Inc. (AMAT) *Real-Time Dispatcher (RTD)* auf. Der RTD verfügt über ein eigenes Daten *Repository*, das jedoch aus der Stammdaten-Datenbank (DB) und der Produktions-DB gespeist wird. Funktional nimmt das *Repository*

keine gesonderte Rolle ein und ist nur durch die *Software*-Implementierung des RTD bedingt.

Im RTD sind eine Reihe von Heuristiken hinterlegt, mit denen die Entscheidung getroffen wird. In Kapitel 7.5 wird der RTD näher besprochen. Die verwendeten Heuristiken werden in Kapitel 9.2 diskutiert. Im Kontext dieses Kapitels ist er als *Rule-Engine* ausreichend charakterisiert.

Bei der Anordnung der Aufträge nach ihrer Priorität greift der RTD auf die Ergebnisse des Schedulers zurück. Die Qualität der Ergebnisse des Schedulers ist höher als bei Heuristiken. Allerdings ist die Implementierung eines Optimierungsmodells aufwendig. Zudem limitiert die Rechenzeit den Einsatz auf einzelne Maschinengruppen oder Maschinen.

Der Scheduler erstellt etwa alle 15 Minuten einen Plan für die nächsten ein bis zwei Stunden. Dabei bedient er sich aus den gleichen Datenquellen wie der RTD. Bei dem Scheduler handelt es sich um einen mathematischen Optimierer. Konkret wird das IBM *ILOG CPLEX Optimization Studio* (International Business Machines Corporation (IBM), 2018) eingesetzt. Die Ergebnisse des Optimierers dienen dem RTD als zusätzliche Datenquelle. Er unterzieht sie einer Konsistenz-Überprüfung, in der auch festgestellt wird, ob die bis zu 15 Minuten alten Pläne noch aktuell sind. Mögliche Ereignisse, die ein Abweichen notwendig machen, sind Maschinenausfälle oder das Ankommen eines Eilloses.

Die hybride Lösung aus Heuristiken und Optimierung gilt als Stand der Technik (Ham et al., 2015). Eine Auswertung in einer Lithographie Maschinengruppe bei Intel Corporation ergab, dass die häufigste Frequenz von Ereignissen, die eine Neuberechnung des Optimierers notwendig machen, bei fünf Minuten liegt (Govind et al., 2008). Obwohl Optimierer bessere Lösungen als Heuristiken liefern, ist die Ebene des RTD unumgänglich für eine Echtzeitfähigkeit des Systems (Govind et al., 2008).

Der RTD leitet das Ergebnis des Schedulers, eine Prioriätsliste der Produktionsaufträge an einer Maschine, an das MES weiter. Von dort werden die Daten entweder an das AMHS oder die Benutzerschnittstelle für Werker weitergeleitet. Der Werker führt den Auftrag aus, was über die Buchung an der Maschine im MES und der Produktions-DB erfasst wird. Auf allen Ebenen

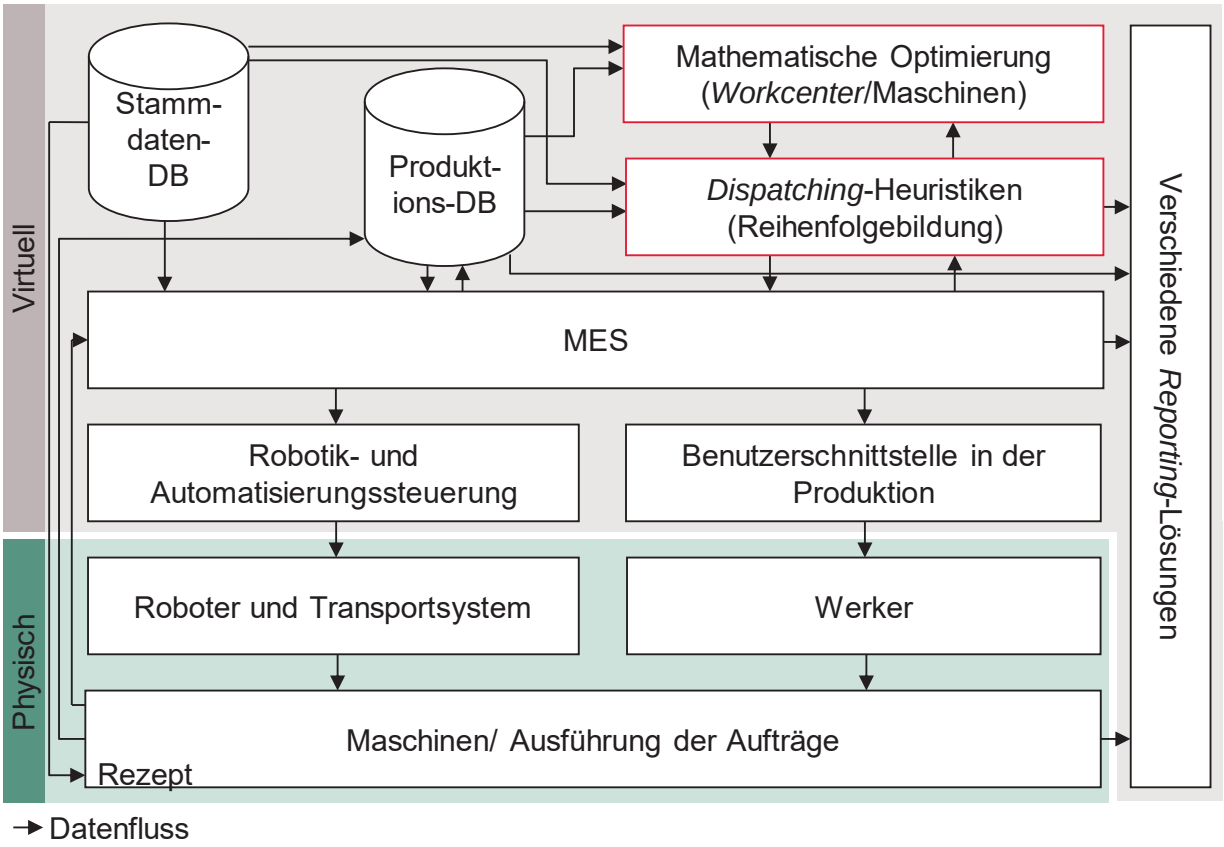


Abbildung 4.8.: Produktionssteuerung in der Infineon Waferprozessierung Dresden. Quelle: Infineon Technologies AG

existieren *Reporting*-Lösungen, um eine Transparenz in der Produktion zu gewährleisten.

In Abbildung 4.8 sind die Systeme und DBs zusammengefasst.

Kapitel 5

Reflexion bestehender Ansätze zur Produktionssteuerung komplexer Werkstattfertigungen

In den bisherigen Kapiteln wurden die Grundlagen der komplexen Werkstattfertigung und ihrer Steuerung gelegt. Der Anwendungszusammenhang und die Praxisrelevanz wurden anhand eines Beispiels dargelegt. Zusätzlich wurden zur Vorbereitung des Lösungsansatzes der Arbeit die Grundlagen des ML besprochen.

In diesem Kapitel werden bestehende Methodiken zur Produktionssteuerung einer komplexen Werkstattfertigung analysiert und bewertet. Dafür werden zunächst Bewertungskriterien in Kapitel 5.1 formuliert. Mittels dieser Bewertungskriterien werden die Methodiken evaluiert. Danach werden relevante Ansätze in Kapitel 5.2 vorgestellt und in Kapitel 5.3 bewertet.

5.1. Bewertungskriterien einer Produktionssteuerung in einer komplexen Werkstattfertigung

Die Manufaktur, die älter als die industrielle Revolution ist, wird primär mit dem Produktionstyp Werkstattfertigungen verknüpft. Obwohl die komplexe Werkstattproduktion in ihrer räumlichen Organisation den gleichen Prinzipien folgt, haben sich die Produktionsart und die Ziele grundlegend geändert. Ursprünglich führte die Werkstattfertigung in der Manufaktur unterschiedliche, unabhängige Gewerke zur Effizienzsteigerung zusammen. Heute liegt der Fokus auf der kundenindividuellen Serien- und Massenproduktion. Dies schlägt sich auch in den Anforderungen an die Produktionssteuerung einer modernen Werkstattfertigung nieder. Diese leiten sich insbesondere aus den Merkmalen einer komplexen Werkstattfertigung ab, die in Kapitel 2.2 vorgestellt wurden. Die Produktionssteuerung muss in einem dynamischen Umfeld die Reihenfolgesteuerung in Echtzeit vornehmen. Dabei muss immer das beste Verhältnis aus Auslastung, DZ und Liefertreue gewährleistet werden. Gleichzeitig sollten die Funktion und die Ziele der Steuerung in der komplexen Umgebung transparent sein. Bei Veränderungen der Ziele oder der Produktion müssen die Änderungen unkompliziert umgesetzt werden können. Die Anforderungen an die Produktionssteuerung einer komplexen Werkstattfertigung sind im Folgenden aufgelistet:

- **Lokale Qualität der Ergebnisse:** In einer komplexen Werkstattfertigung wird die Reihenfolgeplanung lokal an einer Maschinengruppe vorgenommen. Primär wird daher die Planung und Steuerung daran gemessen, wie gut die Zielerreichung lokal gelingt.
- **Echtzeitfähigkeit:** Um auf unvorhergesehene Ereignisse und Änderungen in der Produktion reagieren zu können, sollte die Steuerung in der Lage sein, in Echtzeit umzudisponieren.
- **Globale Optimierung:** Bei der dezentralen Entscheidungsfindung in der Produktionssteuerung komplexer Werkstattfertigungen muss sichergestellt werden, dass globale Ziele, beispielsweise *Key Performance Indicators (KPIs)* der Fabrik, erreicht werden. Die lokale Optimierung ei-

ner Maschinengruppe kann an einer nachfolgenden Maschinengruppe in der Produktionskette den optimalen Betriebspunkt durch die Zusammensetzung der angelieferten Aufträge verhindern. Der Durchsatz der Produktionskette ist durch den Flaschenhals limitiert.

Beispiel: Kapazitätsverluste durch dezentrale Optimierung in der Halbleiterindustrie

Die Lithographie ist in der Halbleiterfertigung ein entscheidender Schritt: Die Maschinengruppe wird für jeden *Layer* jedes Produkts benötigt. Zudem gehören Lithographieanlagen zu den teuersten Anlagen in einer Halbleiterfertigung.

Während der Dotierung des Wafers steht die Lithographie in einer Produktionskette mit der Ionenimplantation. Durch den wiedereintretenden Fluss kommt es zu einer Schleifenbildung, in der sich die Lithographie- und Implantations-Maschinengruppe – mit vernachlässigbaren Zwischenschritten – gegenseitig beliefern. Die Implantation arbeitet rüstoptimiert. Die Implantation bevorzugt daher die Abfertigung von mehreren Aufträgen des gleichen Produkts hintereinander. In der Lithographie ist in den meisten Fällen für eine Produktebene nur eine Fotomaske verfügbar, weshalb eine EPA nicht parallel auf mehreren Maschinen ausgeführt werden kann.

Der Durchsatz einer Implantationsanlage liegt typischerweise über der eines Lithographieclusters. Daher sind in einer Produktion mehr Lithographie- als Implantationsanlagen vorhanden, um an beiden Maschinengruppen den gleichen Durchsatz zu erreichen.

Die lokale Optimierung sowohl an der Lithographie- als auch Implantations-Maschinengruppe führt zu einem globalen Suboptimum. Arbeitet die Ionenimplantation vollständig rüstoptimiert, bildet sich in der Lithographie eine Warteschlange. Der Durchsatz

> an einer einzelnen Lithographieanlage ist deutlich geringer als an der Implantation. Gleichzeitig können identische, parallele Lithographieanlagen stillstehen, da nur eine Fotomaske verfügbar ist. Eine lokale Optimierung an der Lithographie führt im Umkehrschluss zu häufigeren Rüstvorgängen an der Implantation.
>
> Das Beispiel deckt ein typisches Problem der dezentralen Steuerung auf: Die lokale Optimierung der Maschinengruppen führt nicht unbedingt zu einem optimalen Ergebnis für die gesamte Produktion. Um den Durchsatz der Produktionskette zu optimieren, muss eine Optimierung beide Maschinengruppen mit einbeziehen.

- **Transparenz in der Auswirkung auf Zielgrößen:** Die meisten Heuristiken geben klare Handlungsanweisungen. Die Transparenz in der Auswirkung auf die Zielgrößen bei Umsetzungen ist jedoch nicht gegeben. Mit dem Bewertungskriterium wird gemessen, wie klar die Zielsetzung in der Steuerungsmethodik erfasst ist und umgesetzt wird.
- **Adaptivität:** Anpassungsfähigkeit an Veränderungen spielt eine Rolle, wenn neue Produktionsanlagen eingebracht werden, sich die Zielgrößen der Produktion verändern oder sich das Produktionsportfolio verschiebt. Dann ist eine Anpassung der Steuerung notwendig. Mit der Adaptivität soll der Aufwand bewertet werden, mit dem diese Veränderungen implementiert werden können.

Neben den direkten Bewertungskriterien der Methodiken wird ein weiteres Kriterium angewandt, das sich direkt auf die Einführung der Methode in die Produktion bezieht:

- **Kontinuität:** Die meisten existierenden Werkstattfertigungen werden mit Heuristiken betrieben. Mit der Anforderung der Kontinuität wird bewertet, wie kontinuierlich eine neue Produktionssteuerung eingeführt werden kann. Es existiert eine Brücke zwischen der alten und neuen Steuerung. Einzelne Maschinen oder Maschinengruppen können einzeln umgestellt werden.

Unter diesen Bewertungskriterien werden die folgende Methodiken beurteilt.

5.2. Klassische Ansätze der Produktionssteuerung[1]

Die bestehenden Ansätze der Produktionssteuerung einer komplexen Werkstattfertigung werden basierend auf der jeweiligen Optimierungsmethode in fünf Gruppen eingeteilt. Die Optimierungsmethode bezieht sich auf die Art und Weise, wie das spezifische Optimierungsproblem in der Produktionssteuerung gelöst wird. Bei der fünften Gruppe, hybride Systeme in Kapitel 5.2.5, handelt es sich um ein kombiniertes Verfahren aus mathematischer Optimierung und Heuristiken. Aufgrund der breiten Anwendung in der Industrie wird das hybride System separat vorgestellt.

5.2.1. *Dispatching*-Heuristiken

Dispatching-Heuristiken sind die einfachsten und in der Praxis am häufigsten eingesetzten Verfahren (Schuh et al., 2014, S. 215). Es gibt unzählige Varianten, von denen hier nur die wichtigsten vorgestellt werden sollen[2].

- ***First In First Out (FIFO)*:** Die Aufträge werden in der Reihenfolge abgearbeitet, in der sie eintreffen. Es gibt keine Reihenfolgevertauschungen, was die Regel unflexibel macht.
- **Frühester Plan-Starttermin:** Der Auftrag mit dem frühesten Plan-Starttermin erhält in der Produktion die höchste Priorität. Damit wird sichergestellt, dass Aufträge nicht zu lange in der Produktion liegen.
- **Frühester Plan-Endtermin (engl. *Operation Due Date (ODD)*):** Die Regel orientiert sich am Plan-Endtermin der einzelnen Arbeitsschritte und erzielt damit eine hohe Liefertreue. Der veranschlagte Zeitraum t_{EPA} für einen einzelnen Arbeitsschritt, bestehend aus Transport-,

[1] Das Kapitel orientiert sich an Waschneck et al. (2016a).
[2] Die Darstellung folgt der Quelle Schuh et al. (2014, S. 2015 ff.).

Warte- und Bearbeitungszeit, kann über den Flussfaktor berechnet werden:

$$t_{\text{EPA}} = FF_{\text{EPA}} \cdot RPZ_{\text{EPA}}. \tag{5.1}$$

Ausgehend vom geplanten Liefertermin werden dann die Plan-Endtermine der einzelnen Arbeitsschritte berechnet. Vorteil des Vorgehens ist die gleichmäßige Verteilung der Wartezeit auf die Produktionsschritte, die zu einem gleichmäßigen Materialfluss in der Produktion führt.

- **Kürzeste Operationszeit:** Der Auftrag mit der niedrigsten Bearbeitungszeit an der Maschinengruppe hat höchste Priorität. Die Regel reduziert schnell lokal den Bestand.
- **Kürzeste Fertigungszeit:** Der Auftrag mit der niedrigsten noch anstehenden Bearbeitungszeit hat die höchste Priorität. Die Regel reduziert den Bestand in der Linie.
- **Lieferterminregel:** Der Auftrag mit dem frühesten Liefertermin hat die höchste Priorität.
- **Rüstzeitregel:** Der Auftrag mit der geringsten Rüstzeit genießt die höchste Priorität.

Die Liste der Heuristiken lässt sich beliebig erweitern. Heuristiken sind einfach zu entwickeln und für Menschen intuitiv verständlich. Die Implementierung ist einfach, wenn die richtige Regel durch einen Experten festgelegt wurde.

In komplexen Werkstattfertigungen wird meistens eine Hierarchie von mehreren Heuristiken eingesetzt (Fordyce et al., 2015). Außerdem werden für unterschiedliche Maschinentypen unterschiedliche Regeln angewendet (Fordyce et al., 2015). Keine Heuristik zeichnet sich durch konstant bessere Leistung unter verschiedenen Zielgrößen und Szenarien aus (Holthaus et al., 1997; Uzsoy et al., 1993). Stattdessen müssen die Heuristiken für die jeweiligen Zielgrößen und Szenarien gezielt ausgewählt werden.

Die lokale Qualität der Ergebnisse ist gut, aber bei parallelen, identischen Maschinen können Schwierigkeiten auftreten: Die meisten Regeln zielen nur auf ein sofortiges Ergebnis ab und vernachlässigen langfristigen Erfolg. Mathematische Optimierung erzielt meistens bessere Ergebnisse (siehe z.B. Klemmt et al. (2009)).

5.2.2. Multi-Agenten-Systeme

MAS sind ein interessantes Forschungsfeld in der Literatur und Forschung (Mönch et al., 2011), finden jedoch weniger Anwendung in der Praxis. Ein Grund hierfür ist, dass die Kontrolle über selbstorganisierte Systeme gering ist.

Brauer et al. (1998) nutzen bereits einen MAS-Ansatz für *Scheduling* auf mehreren Maschinen.

Die Machbarkeit eines MAS für das *Scheduling* einer kompletten Halbleiterfabrik wurde mit dem System FABMAS demonstriert (Mönch et al., 2003; Mönch et al., 2006a). Auch ein kleineres Beispiel von Yoon et al. (2008) für die Kontrolle der Intel-Minifab (Kempf, 1994) war erfolgreich, wobei die *Scheduling*-Logik in einzelnen Agenten eingekapselt war.

Nahe verwandt mit MAS sind *Ant-Colony-Optimization (ACO)*-Ansätze, metaheuristische Optimierungen, die auf Schwarmintelligenz beruhen. Es existieren Implementierungen für parallele Maschinen (Lin et al., 2013) und parallele *Batching*-Maschinen mit diversen Nebenbedingungen (Jia et al., 2016; Li et al., 2009).

MAS haben die Probleme der verteilten Erfüllung von Nebenbedingungen und der verteilten Optimierung. Während alle Agenten eine Nebenbedingung erfüllen, kann diese durch die Gesamtheit der Agenten verletzt werden. Das Herunterbrechen von Nebenbedingungen und Optimierungszielen stellt eine Herausforderung dar, die sich auf die Transparenz des Systems auswirkt.

5.2.3. Hyperheuristiken und Maschinelles Lernen

Die Auswahl der richtigen Heuristik ist zeitaufwendig und benötigt meistens nach der Einführung in der Produktion eine Feinabstimmung. Hyperheuristiken sind definiert als Methodik für die automatische Selektion oder Erstellung von Heuristiken (Burke et al., 2010) und sind eine Möglichkeit, die Gestaltung von Heuristiken zu automatisieren. Dabei werden auch Techniken des ML eingesetzt. Hyperheuristiken starten von einer Ausgangsmenge gegebener Heuristiken, die neu kombiniert werden.

Eine einfache Variante ist die Superposition unterschiedlicher Heuristiken. Dabei wird die Lospriorität P_{Los}^{h} mit unterschiedlichen Heuristiken h berechnet und dann über eine gewichtete Überlagerung der Prioritäten eine finale Priorität $P_{\text{Los}}^{\text{final}}$ bestimmt:

$$P_{\text{Los}}^{\text{final}} = \sum_{h=1}^{H} w_h \cdot P_{\text{Los}}^{h}, \tag{5.2}$$

wobei w_h der Gewichtungsfaktor der jeweiligen Heuristik ist. Häufig werden jedoch fortgeschrittenere Kombinationsmöglichkeiten genutzt (Branke et al., 2015).

Die Bestimmung von Hyperheuristiken kann einem SL und UL Vorgehen folgen. Bei SL müssen zuerst optimale Reihenfolgebestimmungen ermittelt werden, was häufig eine Herausforderung darstellt. Wenn dies vorliegt, wurden mit NNs in mehreren Publikationen gute Ergebnisse erzielt, z.B. in Branke et al. (2015), Eguchi et al. (1999), El-Bouri et al. (2000) und Weckman et al. (2008). Auch Entscheidungsbäume (Choi et al., 2011) und logistische Regression (Ingimundardottir et al., 2011) wurden eingesetzt.

Wie bei allen Lernverfahren spielt die Selektion der Eingangsparameter, engl. *Feature Selection*, eine wichtige Rolle (Branke et al., 2016). Dabei kommen Branke et al. (2016) zu dem Schluss, dass *Features* in ihrer möglichst grundlegenden Form bereitgestellt werden sollten. Dies deckt sich mit dem Paradigma des DL, immer Rohdaten zu verwenden.

Die lokale Qualität der Ergebnisse ist gut, aber, wie von Mönch et al. (2006b) angemerkt, nicht gleichauf mit mathematischen Optimierern. Der große Vorteil der Hyperheuristiken ist ihre Echtzeitfähigkeit. Häufig werden Zielgrößen direkt in einer Simulation erfasst. Der Aufwand der Implementierung variiert stark nach Ansatz: Wird mit SL gearbeitet, ist das Erzeugen von Trainingsdaten für ein neues, unbekanntes Szenario eine zusätzliche Hürde. Wird direkt in einer Simulation gearbeitet, läßt sich das Versprechen der automatischen Generierung von Heuristiken einlösen.

Meta-Heuristiken sind Heuristiken, die selbst dem Auffinden von Heuristiken dienen. Sie garantieren nicht die optimale Lösung, sind jedoch häufig äußerst effizient. Genetische Algorithmen, eine Form des ML, gehören zu dieser Kategorie. Inspiriert von biologischer Evolution werden, ausgehend von initialen Kandidaten über Rekombination und Mutation, neue Kandidaten erzeugt (Gupta et al., 2006). Eine Fitness-Funktion bewertet die neuen Kandidaten und wählt die besten aus (Gupta et al., 2006).

Eingesetzt wird das Verfahren von Wang et al. (2002) für parallele Batch-Maschinen. Cavalieri et al. (1999) präsentieren eine Anwendung von genetischer Programmierung bei STMicroelectronics N.V., die den bisherigen FIFO-Ansatz in der Produktion deutlich übertrifft.

Genetische Algorithmen können ebenfalls mit Hyperheuristiken kombiniert werden (Pickardt et al., 2013).

Mönch et al. (2006b) bestimmen die Parametrisierung einer vorausschauenden *Dispatching*-Regel mit einem NN. Sie schließen, dass die *Dispatching*-Heuristik mit dem NN zwar 1–2 % schlechter abschneidet als eine optimale Variante, aber der Rechenaufwand deutlich sinkt und damit die Echtzeitfähigkeit erhöht wird.

Eine der ersten direkten Anwendungen von RL auf *Dispatching* und *Scheduling* stammt von Zhang et al. (1995, 1996). Bradtke et al. (1994) liefern wichtige Grundlagen zu kontinuierlichen Markow-Entscheidungsprozessen und demonstrieren Ihre Ergebnisse mit einem System aus zwei heterogenen Maschinen, in dem die Warteschlangenlänge minimiert wird.

Mahadevan et al. (1997) nutzen RL zur Verbesserung des Wartungsplans einer Maschine. Das Model wurde dann konsequent auf die Wartung eines Fließbandes ausgedehnt (Mahadevan et al., 1998).

Riedmiller et al. (1999) nutzen ein NN, um eine *Q*-Funktion in einem RL-System zu repräsentieren. Damit wird ein Szenario mit drei Produktionsressourcen optimiert.

Paternina-Arboleda et al. (2005) kombinieren einen MAS-Ansatz mit RL. Allerdings beschränken sie sich auf eine einzige Maschine, an der unterschiedliche Produkte gefertigt werden. Der Fokus bei dieser Kombination von RL und MAS liegt auf der lernenden Komponente des Systems, weshalb dieser (und weitere Ansätze) in diesem Kapitel vorgestellt werden.

Gabel et al. (2007) nutzen bereits einzelne Agenten an unterschiedlichen Maschinen, bei denen die *Dispatching*-Strategien über RL optimiert werden. Allerdings sind die Maschinen identisch und das Problem statisch.

In neuerer Zeit wurde der Ansatz mit RL durch die großen Fortschritte im Bereich des ML und DL wieder vermehrt untersucht. Qu et al. (2016) greifen den MAS mit RL, genauer *Q*-learning, im Jahr 2016 wieder auf. Sie optimieren eine Produktionslinie mit mehreren Stufen und mehreren Produkten. Allerdings werden keine NNs für die Repräsentation der *Q*-Funktion genutzt.

In einem kurzen Beitrag stellen Kim et al. (2017) eine Anwendung des *Actor-Critic*-Verfahrens des RL für *Dispatching* vor. Die Veröffentlichung bleibt in einigen Bereichen, beispielsweise der Formulierung des Zustandsraums, leider vage.

Zhang et al. (2017) präsentieren einen RL-Ansatz für eine stark vereinfachte Fertigungslinie. Das NN für die Repräsentation der *Q*-Funktion besitzt jedoch nur 18 Eingangsneuronen und eine verdeckte Ebene mit fünf Neuronen. Im Vergleich mit Heuristiken erzielt der adaptive Ansatz konstant bessere Ergebnisse in unterschiedlichen Szenarien.

Stricker et al. (2018) optimieren eine kleine Halbleiterlinie mit einem RL-Ansatz, in dem eine *Q*-Funktion durch ein NN approximiert wird. Die Halbleiterlinie ist stark vereinfacht. Im Benchmark wird eine reine FIFO-

Steuerung in der Auslastung um 8 % übertroffen. Allerdings kann FIFO nicht als Stand der Technik betrachtet werden.

Das Feld der Anwendungen des ML ist äußerst breit und entwickelt sich schnell weiter. Es existieren viele inhomogene Ansätze mit unterschiedlichen Stärken und Schwächen. Meistens werden die Ansätze an relativ kleinen statischen Beispielen validiert und erreichen in diesen ausgesuchten Fällen eine gute Leistung. In der Industrie spielen die Ansätze jedoch keine Rolle: Sie sind zu experimentell und können häufig nicht auf größere Szenarien skaliert werden. Bei der Defizitbewertung in Kapitel 5.3 liegt der Fokus auf den Hyperheuristiken und Meta-Heuristiken, wobei stabile Ansätze des ML in der Produktionssteuerung ebenfalls mit eingehen.

5.2.4. Mathematische Optimierung

Die mathematische Optimierung, entwickelt von Euler[1] und Lagrange[2] (Tichatschke, 2008), ist deutlich älter als die industrielle Fertigung. Kantorovich[3] führte 1939 die Methode der linearen Optimierung oder linearen Programmierung ein (Tichatschke, 2008). Seit über 60 Jahren wird sie genutzt, um *Scheduling*-Probleme in der Industrie zu lösen (Muth et al., 1963). Aber erst der starke Anstieg in der Rechenleistung von Computersystemen machte die Methode in den letzten zehn Jahren wirklich interessant für die Industrie. Trotzdem herrschen einfachere, nicht-exakte Verfahren in der Industrie weiterhin vor (Pinedo, 2016, S. 375).

Meistens werden die Probleme in den Programmierparadigmen *Linear Programming (LP)*, spezifischer meist *Mixed Integer Linear Programming (MILP)*, *Constrained Programming (CP)* und *Dynamic Programming (DP)* ausgedrückt. Kommerzielle Software – wie der Gurobi Optimizer (Gurobi Optimization, Inc, 2015), IBM ILOG CPLEX (International Business Machines Corporation (IBM), 2018) und FICO XPress Optimization (Xpress, 2018) – wird genutzt, um die Probleme zu lösen. Die genannten Optimierer

[1] Leonhard Euler (*1707–†1783)
[2] Joseph Louis Lagrance (*1736–†1813)
[3] Leonid Vitalevich Kantorovich (*1912–†1986)

zeigten alle in den letzten Jahren einen starken Anstieg in der Leistung (Ku et al., 2016; Meindl et al., 2012).

Trotz der Fortschritte bleibt die mathematische Tatsache bestehen, dass das Problem in der Komplexitätsklasse „Nichtdeterministische Polynomialzeit (NP)“ liegt. Daraus resultiert ein exponentielles Skalieren der Rechenzeit mit der Problemgröße. Durch Näherungen und Heuristiken kann die Rechenzeit verkürzt werden, garantiert dann aber kein optimales Ergebnis mehr. Diese Eigenschaft limitiert den Einsatz auf kleine Problemgrößen, normalerweise eingegrenzt auf eine Maschinengruppe.

Ein Ansatz, um mit dieser Einschränkung umzugehen, sind Dekompositionsmethoden. Dabei wird ein Problem in eine Reihe kleinerer Unterprobleme eingeteilt. Die Probleme werden einzeln gelöst und dann wieder zu einer Lösung des vollständigen Problems zusammengesetzt (Klemmt, 2012, S. 117 ff.; Ovacik und Uzsoy 1997). Dabei kann eine Dekomposition nach unterschiedlichen Kriterien erfolgen (Ovacik et al., 1997):

- zeitliche Dekomposition,
- Dekomposition nach Maschinengruppen,
- Dekomposition basierend auf Eigenschaften der Aufträge,
- Dekomposition basierend auf Eigenschaften der Operationen.

Mit der Dekomposition des Problems ist in den meisten Fällen jedoch nicht mehr garantiert, dass die optimale Lösung gefunden werden kann.

Oftmals sind die Ergebnisse eines Optimierers nicht intuitiv verständlich. Komplexe Lösungen erfordern generell, dass sich Werker stärker an den Vorgaben des Systems als an eigenen Abläufen bzw. Gewohnheiten orientieren (Branke et al., 2016). Weil die Ergebnisse nicht ohne weiteres verständlich sind, ist außerdem ein hohes Maß an Vertrauen in das System notwendig (Branke et al., 2016).

Da eine Minimierungsfunktion definiert wird, ist das Verfahren direkt an der Zielgröße und nicht am Ablauf orientiert. Dies führt einerseits zu guten Ergebnissen, andererseits sind Ergebnisse meistens für Werker nicht intuitiv umsetzbar.

Aufgrund der Einschränkungen werden mathematische Optimierungsverfahren meistens mit *Dispatching*-Heuristiken in einem hybriden Verfahren kombiniert. Darauf wird näher in Kapitel 5.2.5 eingegangen.

5.2.5. Hybride Systeme aus Heuristiken und mathematischer Optimierung

In der Literatur existieren diverse Hybridverfahren, bei denen unterschiedliche Methoden des *Dispatching* und *Scheduling* kombiniert werden. Allerdings findet nur ein Ansatz weite Anwendung in der Industrie: Der hybride Ansatz aus Heuristiken und mathematischer Optimierung wird in der Halbleiterindustrie als das optimale Zielsystem angesehen (Govind et al., 2008; Ham et al., 2015; Hanny, 2011). Daher beschränkt sich die Diskussion der Hybridansätze auf die Kombination von Heuristiken und mathematischer Optimierung.

Es handelt sich dabei um eine Kombination äußert unterschiedlicher Ansätze: Heuristiken sind der einfachste Ansatz bezüglich Rechenzeit und Implementierung. Sie liefern jedoch – insbesondere bei mehreren Zielgrößen und Nebenbedingungen – die schwächsten Ergebnisse. Die mathematische Optimierung ist kompliziert zu implementieren, erfordert immer Fachexperten und hat lange Rechenzeiten. Dafür garantiert sie das optimale Ergebnis. Daher wird die Produktion prinzipiell über Heuristiken gesteuert, die an Flaschenhälsen in der Produktion auf Prioritätsbestimmungen von Optimierern zurückgreifen.

Nachteil ist, dass zwei Systeme gepflegt werden müssen. Die Implementierung und Änderungen an der mathematischen Optimierung sind kompliziert. Der lokale Einsatz von Optimierern führt zu keiner übergreifenden bzw. globalen Koordination zur Zielerreichung.

Die Kombination mit den Heuristiken erleichtert jedoch die Einführung von mathematischer Optimierung in der Produktion erheblich. Das System muss nicht von Tag 1 an reibungslos funktionieren, sondern verfügt über eine bewährte Rückfallebene. Über die Kontinuität in den Lösungen sagt dies jedoch nichts aus und das Ergebnis kann von Fall zu Fall äußert unterschiedlich sein.

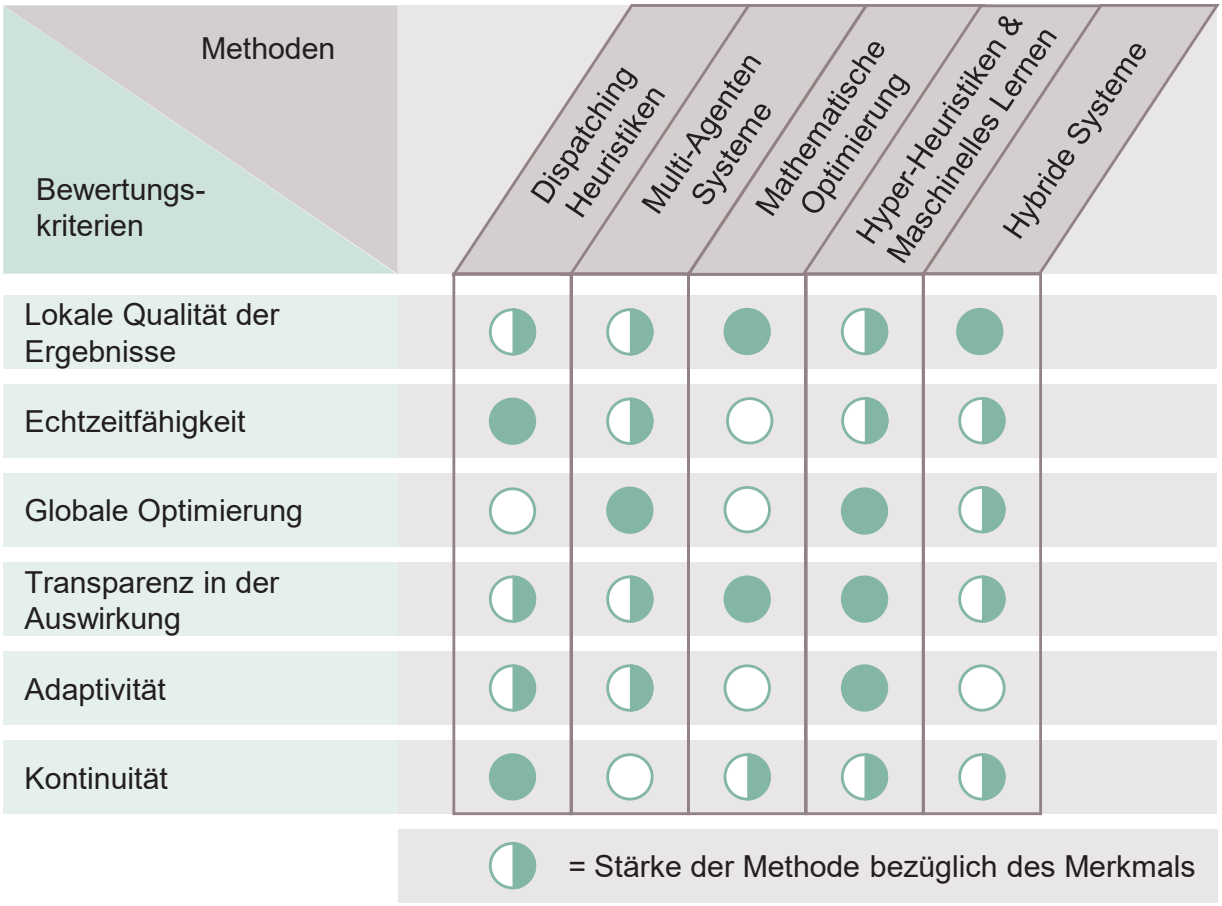


Abbildung 5.1.: Bewertung von Ansätzen zur Produktionssteuerung einer komplexen Werkstattfertigung in einer Bewertungsmatrix nach den Kriterien aus Kapitel 5.1.

5.3. Zwischenfazit und Defizitbewertung

Keine der existierenden Methoden erzielt unter allen Bewertungskriterien aus Kapitel 5.1 gute Ergebnisse und erfüllt damit die Anforderungen, die eine komplexe Werkstattfertigung an eine Produktionssteuerung stellt. Abbildung 5.1 fasst die Defizitbewertung zusammen.

Alle Methoden haben Stärken und Schwächen, wobei sich keine konstant als gut herausstellt. Dies begründet den Einsatz von hybriden Methoden in der Industrie, wobei diese durch zwei Methoden und Systeme einen zusätzlichen Aufwand und eine gesteigerte Komplexität erzeugen. Des Weiteren sinkt die Transparenz in hybriden Systemen.

ML stellt einen interessanten und relativ neuen Ansatz dar. In diesem Bereich existieren vielversprechende Arbeiten, die sich meistens auf einzelne Aspekte konzentrieren: Die Ansätze werden in relativ kleinen Simulationsszenarien getestet.

Die Leistungsfähigkeit des DL hat in den letzten Jahren rasante Fortschritte gemacht (LeCun et al., 2015). Die Kombination mit neuen RL-Agenten ergab neue Algorithmen, die vormals für Computer unlösbare Probleme machbar werden ließen. Das bekannteste Beispiel ist das asiatische Brettspiel *Go*. Während im Schach bereits 1997 der amtierende Weltmeister Garry Kasparov durch den IBM Supercomputer Deep Blue geschlagen werden konnte (Hsu, 2004, S. 3 ff.), waren menschliche Spieler in *Go* lange unschlagbar. Erst 2016 konnte ein Softwareprogramm einen der weltbesten Spieler, den Koreaner Lee Sedol, schlagen (Silver et al., 2016). Die eingesetzte Software, *AlphaGo*, basierte auf DL und RL (Silver et al., 2016).

Auch im Ressourcenmanagement konnten diese Algorithmen erfolgreich angewendet werden und schlugen dort bekannte Heuristiken (Mao et al., 2016).

Mnih et al. (2015) publizierten den DQN-Agent für RL, der innerhalb kürzester Zeit jegliche Art von Atari-Spielen selbst erlernte. Die Leistungsfähigkeit des DQN-Agenten übertraf bestehende Ansätze deutlich. Im folgenden Kapitel wird eine Produktionssteuerung für komplexe Werkstattfertigungen entwickelt, die auf mehreren DQN-Agenten basiert. Durch die gesteigerte Leistungsfähigkeit eines RL-Agenten ist es möglich, eine Methodik für die Produktionssteuerung zu entwickeln, welche die besprochenen Anforderungen erfüllt. Nach der Vorstellung des Lösungsansatzes und der Validierung im Fallbeispiel wird die Leistungsfähigkeit der neuen Methodik in Kapitel 8.4.1 in Bezug auf die Bewertungskriterien beurteilt.

Kapitel 6

Lösungsansatz: Methodik zur autonomen Produktionssteuerung

Die Arbeit kombiniert einen RL-Ansatz mit einem MAS für eine neuartige Produktionssteuerung für komplexe Werkstattfertigungen. Mehrere DQN-Agenten optimieren die Reihenfolgebildung an Anlagen, um beispielsweise Auslastung oder Termintreue zu verbessern. Es werden Algorithmen des DL genutzt, die aus Daten und Erfahrungen in der Simulation lernen, die auf dem Digitalen Zwilling beruhen. Damit wird die Industrie 4.0-Vision der selbstorganisierten, autonomen und dezentralen Produktionssteuerung in komplexen Werkstattfertigungen umgesetzt (Müller et al., 2016, S. 9 und 12).

Zuerst wird in diesem Kapitel die Fabriksimulation beschrieben, auf der der RL-Ansatz basiert. In Kapitel 6.2 werden der Zustands- und Aktionsraum definiert, um den Ablauf in der Fabrik als Markow-Entscheidungsprozess zu formulieren. Danach wird in Kapitel 6.3 der Kern der neuen Methode vorge-

stellt: ein autonomes *Dispatching*-System, das auf ML basiert. Im nächsten Teil, Kapitel 6.4, wird der MAS-Ansatz eingeführt und begründet. Zuletzt wird in Kapitel 6.5 das Übertragen der angelernten Produktionssteuerung in die Produktion beschrieben.

6.1. Simulationsmodell als Grundlage der Methodik

Insbesondere RL setzt ein Simulationsmodell voraus, das in den entscheidenden Aspekten die Produktion realitätsnah abbildet. Besonders gut geeignet ist ein Digitaler Zwilling, bei dem eine Synchronisation mit der realen Fertigung sichergestellt ist (Talkhestani et al., 2017). Auf die Sicherstellung der Qualität und Synchronisierung des Simulationsmodells wird näher in Kapitel 7.2 eingegangen. Im Folgenden wird angenommen, dass eine hinreichende Übereinstimmung zwischen physischer Realität und Simulation vorliegt, sodass Simulationsergebnisse ohne Weiteres übertragbar sind.

In einer ereignisorientierten Simulation erfolgt die Zeitdiskretisierung über die zeitlich geordnete Sequenz der Ereignisse. Die Zeitabstände zwischen den Ereignissen sind nicht einheitlich, was zu einer Abkopplung der Rechenzeit in der Simulation zu einem simuliertem Zeitraum führt. Lange Zeiträume ohne Ereignisse können dadurch in der Simulation extrem schnell verarbeitet werden.

In der Fabriksimulation werden zwei Arten von Ereignissen unterschieden: Ereignisse, die eine Aktualisierung oder Veränderung der Reihenfolgeplanung nötig machen könnten, und solche, die keinen Einfluss auf die Reihenfolgeplanung haben. In der Simulation wird ein Ereignisfilter eingesetzt, der beide Ereignisarten unterscheidet. Im Folgenden werden nur Ereignisse betrachtet, die eine Entscheidung in der Reihenfolgeplanung erfordern können. Als solches Ereignis wird die Ankunft eines neuen Loses betrachtet (engl. *ARRIVAL*) und die Fertigstellung eines Loses an einer Maschine (engl. *MOVEOUT*). Sowohl ein *ARRIVAL*- als auch ein *MOVEOUT*-Event können den Start eines neuen Auftrags auf einer Anlage auslösen.

Durch die *ARRIVAL*- und *MOVEOUT*-Ereignisse wird eine neue Zeitdiskretisierung eingeführt. Die Schritte werden mit Zeitpunkt t bezeichnet, wobei $t \in \mathbb{N}$, da es sich auf einen konkreten Zeitpunkt der Diskretisierung bezieht.

Zu jedem Zeitpunkt t befindet sich das System in einem Zustand $s_t \in \mathcal{S}$, wobei $\mathcal{S}$ der Raum aller möglichen Fabrikzustände ist. In der Fabriksimulation wird dann eine Entscheidung über die nächste Aktion erwartet, damit die Simulation weiterläuft. Diese wird durch die Produktionssteuerung in Form einer Aktion a_t auf Grundlage von s_t ermittelt und der Simulation bereitgestellt. Um die Markow-Eigenschaft im RL zu erfüllen, muss alle relevante Information für die Entscheidung im Zustandsvektor s_t enthalten sein (Sutton et al., 1998, S. 57). Ist die Markow-Eigenschaft erfüllt, definiert die Simulation einen Markow-Entscheidungsprozess. Der zeitliche Ablauf des Markow-Entscheidungsprozesses in der Fabriksimulation ist in Abbildung 6.1 gezeigt.

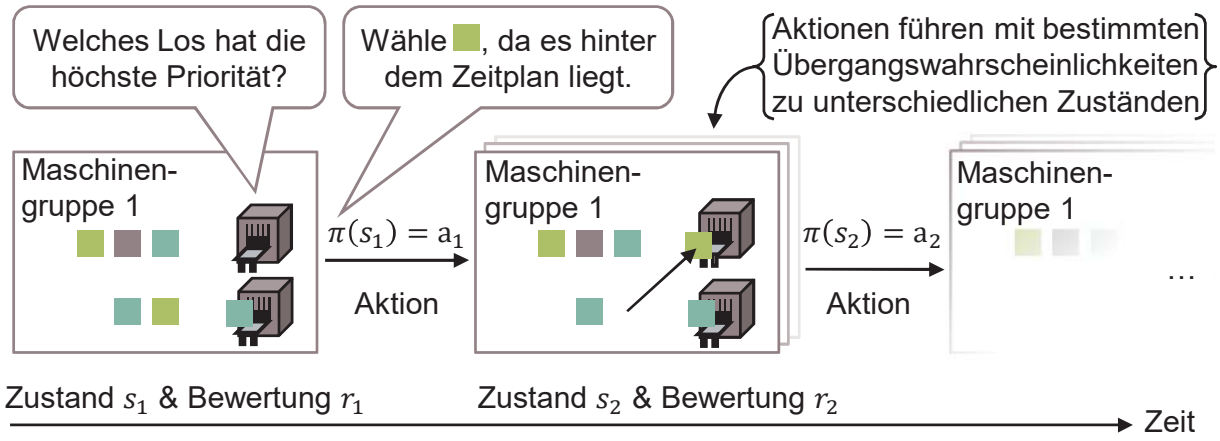


Abbildung 6.1.: Zeitlicher Ablauf des MARKOW-Entscheidungsprozess in der Fabriksimulation am Beispiel einer Maschinengruppe. Ein Ereignis an einer Maschinengruppe erfordert eine *Dispatching*-Entscheidung durch die *Policy* π. Die Ausführung der Entscheidung a_1 führt zu einem neuen Zustand. Der Übergang ist nicht deterministisch, sondern stochastisch, da die gleiche Aktion zu unterschiedlichen Zuständen führen kann. Stochastische Elemente im Übergang sind unter anderem die variierende RPZ und die Ausfallwahrscheinlichkeit von Maschinen.

6.2. Definition des Zustands- und Aktionsraums

Für die Optimierung während des RL ist es entscheidend, dass die optimale Aktion aus der kodierten Information im Zustandsvektor ermittelt werden kann. Im Folgenden werden Zustands- und Aktionsraum definiert. Der Zustandsraum $S = S_{\text{MG}} \otimes S_{\text{Aufträge}}$ ist ein Tensorprodukt der Maschinenzustände einer Maschinengruppe (MG) und der Aufträge. Der Raum der Maschinenzustände $S_{\text{Maschinen}}$ setzt sich aus den Zuständen der einzelnen Maschinen zusammen:

$$\mathbf{s}_{\text{Maschinen}} = \left\langle \mathbf{s}_1 \otimes \ldots \otimes \mathbf{s}_{M_w} \right\rangle \in S_{\text{MG}} = \underbrace{S_{\text{M}} \otimes \ldots \otimes S_{\text{M}}}_{M_w}, \tag{6.1}$$

wobei die Maschinengruppe w aus M_w einzelnen Maschinen besteht. Anschließend wird der Zustandsraum S_M einer Einzelmaschine definiert. Prinzipiell muss der Zustandsraum einer Maschine S_M alle für ihre logistische Verfügbarkeit relevanten Informationen über eine Maschine erfassen. Die relevanten Eigenschaften des Zustands der Maschine sind die dedizierten Prozesse auf der Maschine, die Verfügbarkeit und die aktuelle Rüstung. Die dedizierten Prozesse auf einer Maschine können mit der Dedizierungsmatrix der Maschinengruppe d bestimmt werden. Dabei wird der Maschinenindex der Matrix auf die entsprechende Maschine gesetzt und der resultierende Vektor verwendet. Die Verfügbarkeit v_m einer Maschine m wird als rein binär angenommen, $v_m \in \{0, 1\}$. Die Rüstung r_m kann als sogenannter *One-Hot*-Vektor ausgedrückt werden. Bei RT unterschiedlichen Rüstungen gilt $r_m \in \{0, 1\}^{RT}$. Für einen *One-Hot*-Vektor gilt immer $\sum_{i=1}^{RT} \left(r_m\right)_i = 1$. Dies ist gleichbedeutend zu der Festsetzung, dass eine Maschine nur einen eindeutigen Rüstzustand hat. Der Maschinenzustand setzt sich dann aus diesen drei Elementen zusammen:

$$\mathbf{s}_m = \left\langle d'_{EPA,m}; v_m; r_m \right\rangle. \tag{6.2}$$

Beispiel: Zustandsvektor einer Maschine

Im folgenden Beispiel wird eine Maschinengruppe mit drei Maschinen mit jeweils zwei Rüstungen und drei EPAs betrachtet, die an der Maschinengruppe ausgeführt werden können. Zur Illustration wird der Maschinenzustandsvektor der ersten Maschine zusammengesetzt.

Die Dedizierungsmatrix $d_{EPA,m}$ ist wie folgt belegt:

$$d = \begin{array}{c} \begin{matrix} M_1 & M_2 & M_3 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \end{array} \begin{array}{l} \\ EPA_1 \\ EPA_2 \\ EPA_3 \end{array}. \tag{6.3}$$

Die erste Maschine M_1 kann also die Prozesse EPA_1 und EPA_2 ausführen, die dritte EPA jedoch nicht. Der erste Teil des Zustandsvektors, der die Eigenschaften der Maschine beschreibt, lautet also $\mathbf{s}_{M_1} = \langle d'_{EPA,M_1}, \ldots \rangle = \langle 1, 1, 0; \ldots \rangle$. Danach kommt die binäre Maschinenverfügbarkeit. Im Beispiel wird angenommen, dass die Maschine verfügbar ist, also $v_{M_1} = 1$. Von den zwei verfügbaren Rüstungen ist die erste gewählt, also $r_{M_1} = \langle 1, 0 \rangle$. Der Zustandsvektor der Maschine lautet dann:

$$\mathbf{s}_{M_1} = \langle 1, 1, 0; 1; 1, 0 \rangle . \tag{6.4}$$

Der zweite Teil des Zustandsraums wird von den Aufträgen $S_{\text{Aufträge}}$ gebildet. Hierfür wird zuerst der Zustandsraum für einen Auftrag beschrieben, der sich aus den logistischen Eigenschaften des Auftrags zusammensetzt. Als Erstes wird der Produkttyp pt, der gleichbedeutend zu einem Arbeitsplan ist, als *One-Hot*-Vektor angegeben. Bei p unterschiedlichen Produkttypen gilt $pt \in \{0, 1\}^p$. Der aktuelle Ort o wird ebenfalls als *One-Hot* kodiert, wobei mögliche Orte diskret sind. Ein Ort kann sich auf ein Regal, also ein Lager, das Transportsystem bzw. dedizierte Abschnitte des Transportsystems oder eine Maschine beziehen. Als Nächstes wird der relative Abarbeitungsgrad aa des Arbeitsplans angegeben. Aufgrund der enormen Länge eines Arbeitsplans, in der Halbleiterfertigung teilweise über 1000 Schritte, wird der aktuelle Schritt durch die Länge des Arbeitsplans geteilt.

Zuletzt wird eine relative Abweichung dd eines geplanten Zeitstempels für die entsprechende Operation verwendet. Beim finalen Lieferdatum der Produktion muss die Information mit dem Abarbeitungsgrad und Produkttyp kombiniert werden, um abzuschätzen, wie der Auftrag in der Zeit liegt. Um hier die Komplexität für das NN zu reduzieren, wird mit der Zeitabweichung vom Planungszeitstempel des Auftrags gearbeitet, wobei die Planungszeitstempel mit einem fixierten FF errechnet werden. Bei den über den FF errechneten Planungszeiten wird implizit das Paradigma verwendet, dass eine gleichmäßige Abarbeitung bzw. ein gleichmäßiger Fluss eines Werkstücks in der Produktion erstrebenswert sei. Für die wartenden Lose werden die

Zeitabweichungen dann normalisiert, wobei die größte negative Abweichung auf eins gesetzt wird und die geringste Abweichung auf null.

Der Auftragsvektor eines Produkts setzt sich dann zusammen aus

$$\mathbf{s}_{\text{Auftrag}} = \langle pt; o; aa; dd \rangle . \tag{6.5}$$

Beispiel: Zustandsvektor für einen Auftrag

Im Folgenden wird der Zustandsvektor für ein Los in einem Lager an einer Maschinengruppe beschrieben. Das Los hat den Produkttyp 2 von drei Produkttypen in der Produktion: $pt = \langle 0, 1, 0 \rangle$. Die zwei Lager und zwei Maschinen in der Maschinengruppe werden als Orte 1 bis 4 bezeichnet. Das Los befindet sich in Ort 2, also dem zweiten Lager, womit der Ort als $pt = \langle 0, 1, 0, 0 \rangle$ angegeben wird. Der Arbeitsplan des Loses umfasst 300 Schritte, von denen der aktuelle Schritt der 113te ist. Damit ist der Abarbeitungsgrad $aa = 113/300 = 0,38$.

Insgesamt warten an der Maschinengruppe um 12:17 Uhr drei Lose. Die geplanten Prozessstarts der Lose sind 11:55 Uhr, 12:14 Uhr und 12:31 Uhr. Die relativen Abweichungen betragen damit −22 Minuten, −3 Minuten und +14 Minuten. Die Zeitspanne beträgt damit 36 Minuten. Der Beispiel-Auftrag ist der zweite der drei Aufträge (-3 Minuten Abweichung):

$$dd = \frac{14\text{min} - (-3\text{min})}{36\text{min}} = 0,47. \tag{6.6}$$

Der Zustand des Loses im Zustandsraum lautet damit

$$\begin{aligned} \mathbf{s}_{\text{Auftrag}} &= \langle pt;\ o;\ aa;\ dd \rangle \\ &= \langle 0, 1, 0;\ 0, 1, 0, 0;\ (0,38);\ (0,47) \rangle . \end{aligned} \tag{6.7}$$

Für den vollständigen Zustandsraum der Aufträge ist entscheidend, welche Aufträge für den Agenten sichtbar gemacht werden. Hierfür wird auf die räumliche Anordnung der Anlagen und Lagerpunkte zurückgegriffen. Wird an einer Anlage eine Entscheidung benötigt, wird zuerst nach Lagerorten

gefiltert, die für den Werker oder Roboter erreichbar sind (oder deren Distanzen als vertretbar angesehen werden). Das Transportsystem stellt sicher, dass Lose an die entsprechenden Stellen geliefert werden. Der Zustandsraum baut sich damit zuerst aus den Aufträgen an qualifizierten Lagerorten auf. Danach folgen die aktiven Aufträge auf Maschinen, von denen potentiell an die Maschinengruppe geliefert wird. Damit wird sichergestellt, dass der DQN-Agent eine Möglichkeit hat, seinen zukünftigen Umlaufbestand abzuschätzen.

Da der Umlaufbestand an einer Maschinengruppe variiert, die Anzahl der Eingaben eines NNs jedoch konstant gehalten werden muss, wird eine maximale Anzahl *pos* an Aufträgen pro Maschinengruppe festgelegt. Diese kann der vorhandenen Lagerkapazität in Regalen entsprechen. Wird die maximale Kapazität unterschritten (was meistens der Fall ist), wird der Zustandsvektor mit Nullen aufgefüllt.

Die Reihenfolge, mit der die Aufträge an einer Maschinengruppe dem NN übermittelt werden, wird bei jeder Bestimmung des Zustandsvektors zufällig ermittelt. Dies verhindert ein *Overfitting*, bei dem systematisch solche Positionen in der Warteschlange häufiger ausgewählt werden. Bei den meisten Bewertungsfunktionen haben Aufträge mit langen Wartezeiten eine höhere Priorität. Würden die Aufträge beispielsweise in der Reihenfolge ihrer Ankunft an den DQN-Agent übermittelt, gäbe es Positionen in der Reihenfolge, die eine höhere Wahrscheinlichkeit haben, ein gutes Ergebnis zu erbringen. Damit würde das NN systematisch Positionen bevorzugen, unabhängig von den Eigenschaften der entsprechenden Aufträge.

Die maximale Anzahl *pos* der Aufträge an einer Maschinengruppe bestimmt die Größe des Aktionsraums. Der Aktionsraum bietet $pos + 1$ Aktionen. Die ersten *pos* Aktionen verweisen jeweils auf das Los, das auf der anfragenden Maschine gestartet werden könnte. Die letzte Aktion ist eine Warte-Aktion, bei der kein Auftrag gestartet wird. Dies kann beispielsweise vorkommen, wenn noch auf einen Auftrag gewartet wird, um einen Batch zu vervollständigen.

6.3. Überwachtes und bestärkendes Lernen in der Fabriksimulation

Im Folgenden wird der Kern des autonomen *Dispatching*-Systems vorgestellt. Prinzipiell ist der Ansatz in der Lage, autonom und ohne menschliches Vorwissen, also als reines RL-System, neue *Dispatching*-Strategien zu finden. Trotzdem gibt es drei Gründe, weshalb es sinnvoller sein kann, die eingesetzten NNs zuerst überwacht oder mithilfe der konventionellen Steuerung lernen zu lassen:

1. **Validierung.** Validierung bezieht sich sowohl auf die Topologie des NNs, auf den Zustands- und Aktionsraum als auch auf die Implementierung. Von einer bestehenden Steuerung werden Aktionen zu bestimmten Zuständen in der Simulation generiert. Damit das RL funktioniert, muss das NN prinzipiell in der Lage sein, die Aktionen aus den Zuständen zu konstruieren. Wenn dies mit generierten Zuständen in einem SL-Prozess nicht funktioniert, wird auch ein RL-System scheitern. Allerdings ist SL einfacher zu implementieren und lernt schneller, weshalb es sich besser zur Validierung eignet. Gleiches gilt für die Implementierung in einer Simulation, die Zustände und Aktionen erzeugt. Auch diese lassen sich mit einem geringen Aufwand in SL testen.

2. **Kontinuität.** Durch SL können bestehende Heuristiken durch die NNs erlernt werden. Dadurch wird eine bestehende, regelbasierte Steuerung in NNs übertragen. Während in einem System aus Heuristiken Regeln ausgewählt und hierarchisch angeordnet werden, ist ein NN ein kontinuierliches System, das in einem kontinuierlichen Raum weitaus komplexere Kombinationen erlaubt.

   Nach dem Anlernen durch SL kann die Steuerung durch NNs durch RL weiter optimiert werden. Im RL kann eine kleine Lernrate gewählt werden, sodass ein konstanter Übergang von alter zu neuer Steuerung zu beobachten ist. Durch einen frühzeitigen Abbruch des Lernens kann die Steuerung in kleinen Schritten verändert werden. Das Vorgehen

erzeugt bei Werkern und Fabrikleitern ein größeres Vertrauen in die „Black-Box-Modelle“.

3. **Rechenzeit.** ML-Anwendungen benötigen im Allgemeinen eine hohe Rechenleistung. Da RL-Modelle meistens ohne Vorwissen starten, dauert der Lernvorgang länger. Das vorhandene Vorwissen kann genutzt werden, um die NNs mit RL anzutrainieren und das Training abzukürzen.

Das Übertragen von erlerntem Wissen von einer Aufgabe auf eine andere, ähnliche Aufgabe wird als *Transfer Learning* bezeichnet (Pratt et al., 1996). Im Kontext dieser Arbeit kann die erste Aufgabe als das Reproduzieren der Ergebnisse einer bestehenden Produktionssteuerung aufgefasst werden und die zweite Aufgabe als das Optimieren der Produktionssteuerung im RL.

Im Forschungsbereich des DL wird *Transfer Learning* meistens enger gefasst: Beim Übertragen eines NNs werden die Gewichte mehrerer, verdeckter Ebenen eingefroren (Géron, 2017, S. 287). An der zweiten Aufgabe wird dann nur noch eine Untergruppe der Ebenen trainiert, während die Gewichte in den eingefrorenen Ebenen konstant gehalten werden. Die Reduktion der freien trainierbaren Gewichte beim *Transfer Learning* führt zu einer Reduktion der Rechenzeit sowie einem geringeren Bedarf an Daten (was im RL eine geringe Anzahl an Schritten in der Simulation bedeutet). Dieses Verfahren kann im autonomen *Dispatching* ebenfalls genutzt werden, um das bestärkende Lernen in der Fabriksimulation abzukürzen.

Im Folgenden wird zuerst kurz das überwachte Lernen der Produktionssteuerung beschrieben, um danach auf das eigentliche RL-Modell einzugehen.

6.3.1. Überwachtes Lernen in der Fabriksimulation

Zustands- und Aktionsraum stellen eine allgemeine, abstrahierte Schnittstelle für eine Produktionssteuerung dar. Sie kann genutzt werden, um die Produktion mit einer konventionellen Steuerung zu betreiben. Dadurch können Zustände, Aktionen und Bewertungen aufgezeichnet werden. Aus den

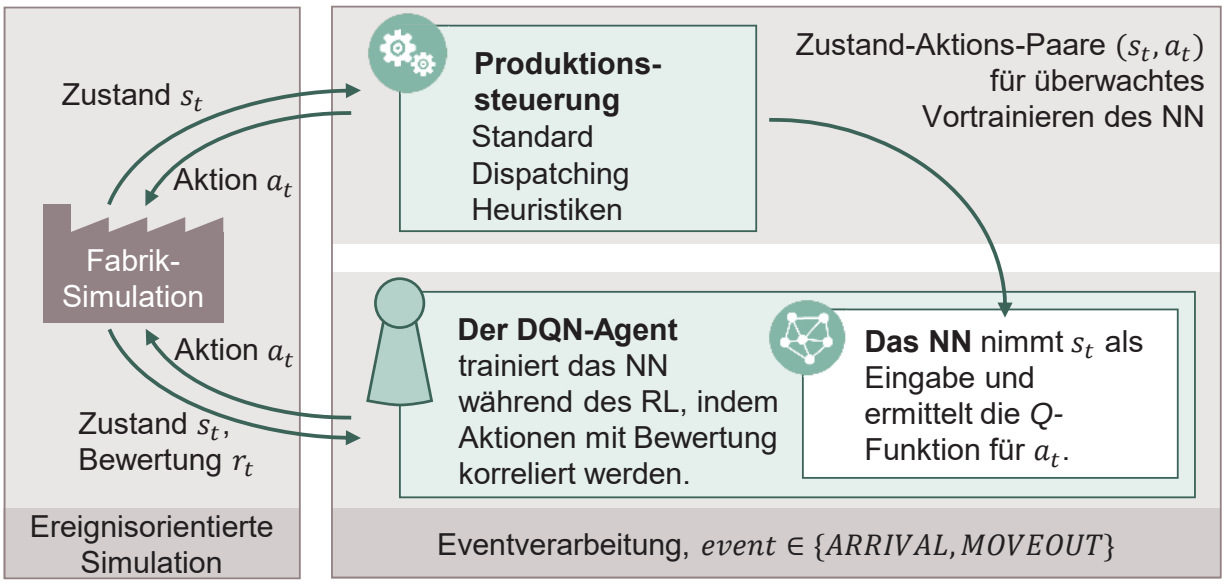


Abbildung 6.2.: Die Fabriksimulation kann entweder durch eine konventionelle Fabriksteuerung oder durch die DQN-Agenten gesteuert werden. Beides erfolgt über einen Markow-Entscheidungsprozess.

Bewertungen können dann Q-Werte für die Ausführung von Aktionen errechnet werden. Die aufgezeichneten Informationen werden genutzt, um das NN zu trainieren. Abbildung 6.2 zeigt das Zusammenspiel von konventioneller Steuerung, Simulation, DQN-Agent und NN .

Das NN bekommt den vollständigen Zustand als Eingabe und sagt die Q-Werte für alle möglichen Funktionen voraus. Beide Informationen liegen nach einem Simulationsdurchlauf mit einer konventionellen Produktionssteuerung vor.

Allerdings wäre dieses NN des DQN-Agenten nur für die Vorhersage der Q-Werte einer konventionellen Steuerung konstruiert. Ein Training mit einer konventionellen Produktionssteuerung erzeugt einen Trainingsdatensatz, der ausschließlich „positive“ Beispiele enthält. Dabei fehlt die Heterogenität im Trainingsdatensatz, in dem keine „negativen“ Beispiele enthalten sind. Da bei einem Zustand immer eine deterministische Aktion durch die Produktionssteuerung ausgewählt wird, würde das NN nur auf diese Aktion trainiert. Die Vorhersagen auf den anderen, suboptimalen Aktionen wären

dementsprechend schlecht, da sie im Trainingsdatensatz nicht vorkommen. Daher sollte zusätzlich zu dem Datensatz mit der konventionellen Produktionssteuerung ein Datensatz mit einer zufälligen Auswahl eines Loses in der Simulation erzeugt werden. Dadurch enthält der Trainingsdatensatz sowohl die Bewertungen von optimalen als auch von suboptimalen Entscheidungen in unterschiedlichen Zuständen.

Nach Abschluss dieses Forschungsprojekts wurde auf der *AAAI*[1] *Conference on Artificial Intelligence* die Publikation *Deep Q-Learning from Demonstrations* (DQfD) veröffentlicht, in der Hester et al. (2018, *Google DeepMind*) den gleichnamigen DQfD-Agenten vorstellen. Der DQfD-Agent lernt von Daten des vorherigen Kontrollsystems. Dadurch wird der Lernvorgang im RL deutlich beschleunigt. Die Integration des DQfD-Agenten in das autonome *Dispatching* verspricht ein vielversprechendes, weiterführendes Forschungsprojekt.

6.3.2. Bestärkendes Lernen in der Fabriksimulation

Mit SL kann eine konventionelle Produktionssteuerung erfasst und imitiert werden. Um die Produktionssteuerung zu verbessern, ist das RL notwendig. Das prinzipielle Zusammenspiel zwischen Umgebung und Agent, das für RL notwendig ist, wurde in Kapitel 3.2 erklärt und für die Fabriksimulation in Abbildung 6.2 gezeigt. Im Folgenden wird der Ablauf inklusive der Schritte in der ereignisorientierten Simulation beschrieben.

Die Fabriksimulation wird solange ausgeführt, bis eine *Dispatching*- Entscheidung an einer Maschine notwendig wird. Im nächsten Schritt wird der Zustandsvektor s_t und die aktuelle Bewertung r_t berechnet. Zusammen mit der Information, welche Maschine die Entscheidung anfragt, werden s_t und r_t an die Eventverarbeitung übergeben (siehe Abbildung 6.2). Danach wird anhand der aufrufenden Maschine ermittelt, welcher DQN-Agent zuständig ist. Die Begründung und Erklärung des MAS wird in Kapitel 6.4 geliefert. Der Lernprozess ist für alle DQN-Agenten identisch.

[1] *Association for the Advancement of Artificial Intelligence (AAAI)*

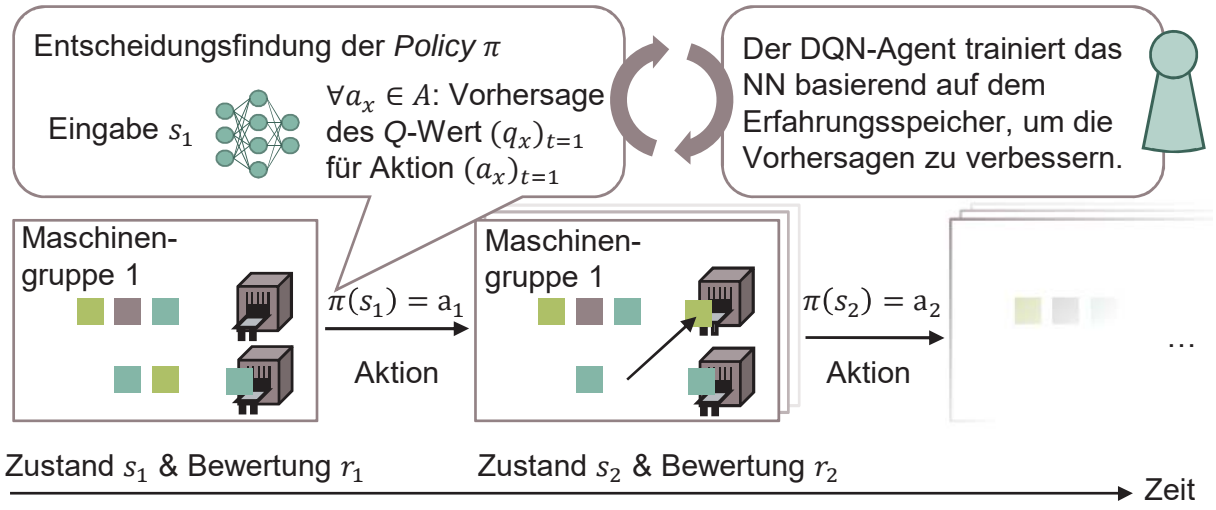


Abbildung 6.3.: Entscheidungsfindung und Training der *Policy* π, die durch das NN repräsentiert wird

Danach zweigt sich der Prozess weiter auf, je nachdem ob sich das System in der Lernphase oder im produktiven Einsatz befindet.

In der Lernphase wird die neue Bewertung genutzt, um die Prognosen des NN – wie in Kapitel 3.2.3 beschrieben – zu verbessern. Danach kommt die sogenannte ϵ-*greedy*-Strategie zum Einsatz. Mit einer Wahrscheinlichkeit von ϵ wird eine zufällige Aktion ausgeführt. Durch diese explorative Aktion lernt der DQN-Agent seine Umgebung kennen und lernt, welchen Einfluss (unbekannte) Aktionen haben. Mit einer Wahrscheinlichkeit von $1 - \epsilon$ wird die Aktion mit dem NN bestimmt. Bei einer *linear annealed* ϵ-*greedy*-Strategie folgt der ϵ Parameter einer linearen Funktion, die mit der Anzahl der ausgeführten Schritten skaliert. Eine Ober- und Untergrenze beschränken die Größe von ϵ. Wenn im NN kein Vorwissen vorhanden ist, empfiehlt es sich, mit einem ϵ-Wert von 1 zu starten, um den Agenten möglichst schnell explorativ die Umgebung erkunden zu lassen. Der ϵ-Wert sollte während der Lernphase nicht unter $0,1 = 10$ % sinken, um ein *overfitting* zu verhindern. Werden alle Aktionen vom NN bestimmt, kann es zufällig passieren, dass eine Aktion permanent nicht vom NN ausgewählt wird. Die schlechte Vorhersage des NNs für die spezifische Aktion ändert sich dann nicht, da sie

nicht explorativ untersucht wird. Dies kann Aktionen dauerhaft verhindern, die in selteneren Zuständen durchaus zum Erfolg führen können.

Um die Aktion mit dem NN zu bestimmen, wird der Zustand s_t dem NN als Eingabe gegeben. In der Ausgabeebene prognostiziert das NN die Q-Funktionen für die möglichen Aktionen. Die Aktion mit der besten Bewertung wird für die Ausführung ausgewählt. Die Aktion wird dann in der Fabriksimulation ausgeführt, die bis zur nächsten *Dispatching*-Entscheidung weiterläuft.

Im produktiven Einsatz springt das System direkt zur Bestimmung der Aktion durch das NN. Es werden keine explorativen Entscheidungen ausgeführt.

Der komplette Prozess ist in Abbildung 6.4 gezeigt. Der zeitliche Ablauf als Markow-Entscheidungsprozess an einer Maschinengruppe ist in Abbildung 6.3 gezeigt.

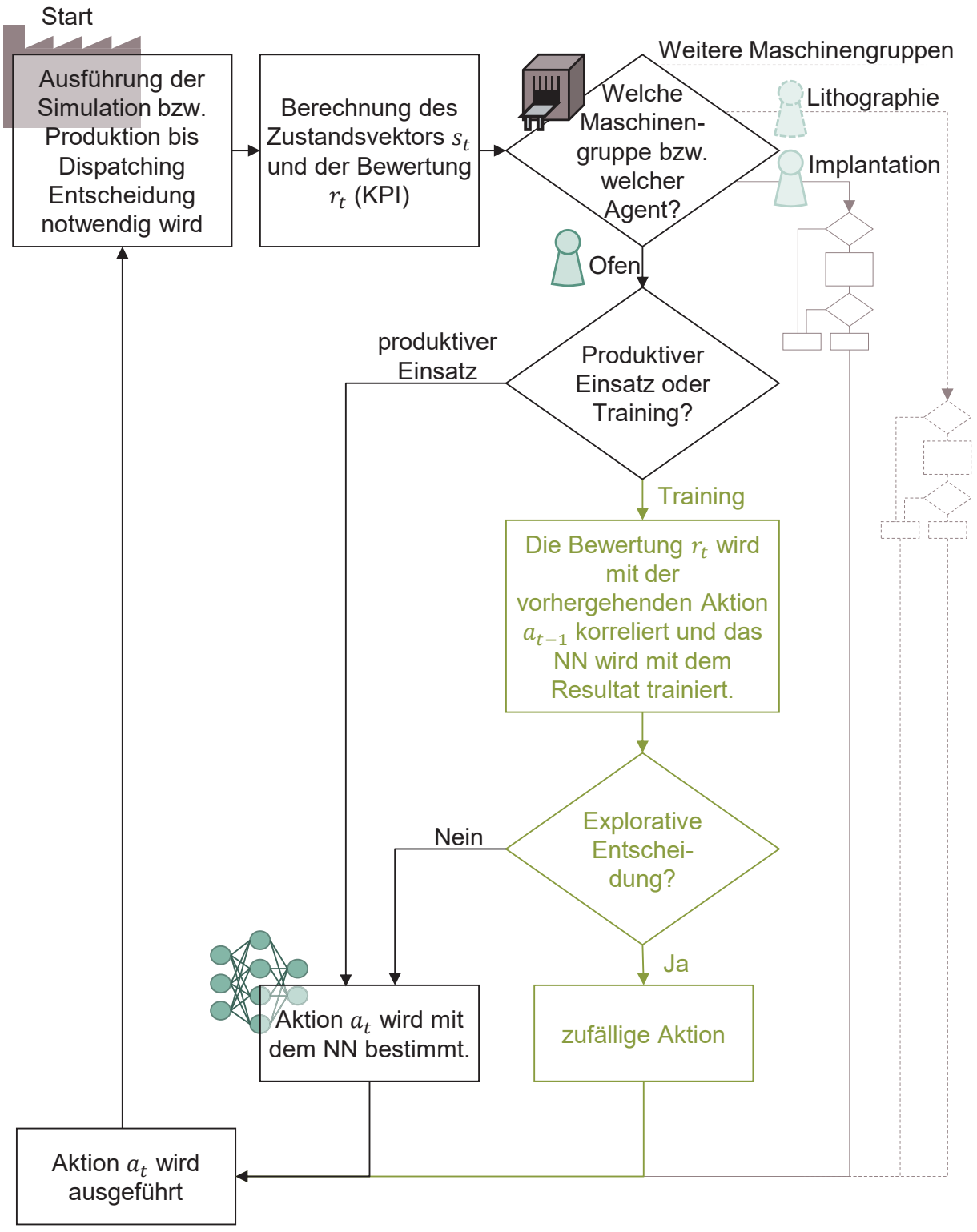


Abbildung 6.4.: Vereinfachter Ablauf des Lernprozesses in der Fabriksimulation

6.4. Multi-Agenten-System in der Produktionssteuerung

In den Experimenten zu dieser Arbeit traten Schwierigkeiten beim Anlernen von mehreren Maschinengruppen durch einen Agenten auf. NNs verarbeiten Informationen besser, wenn diese über den Eingabevektor gleichmäßig verteilt sind. Werden unterschiedliche Maschinengruppen mit dem gleichen Agenten angelernt, muss dieser aufgrund der Information der anfragenden Maschine die komplette restliche Information (vorhandene Aufträge etc.) in einen anderen Kontext stellen.

Um dies zu umgehen, wurde der DQN-Agenten-Steuerungsansatz als MAS konzipiert, bei der jeder DQN-Agent eine Maschinengruppe steuert. Im Folgenden wird beschrieben, wie durch den MAS-Ansatz die Skalierbarkeit und Stabilität in der Lernphase verbessert wird. Dafür wird das Training in zwei Lernphasen eingeteilt, **Phase A** und **Phase B**.

In der ersten Lernphase, **Phase A**, werden die DQN-Agenten unabhängig voneinander trainiert. Dabei wird jeweils nur ein DQN-Agent aktiviert, der eine Maschinengruppe steuert. Die restlichen Maschinengruppen werden von einer konventionellen Steuerung kontrolliert. Der DQN-Agent erhält außerdem eine Bewertungsfunktion, die sich nur auf die Leistung seiner Maschinengruppe bezieht. Das Training der einzelnen DQN-Agenten kann nacheinander oder parallel erfolgen. Dieses Vorgehen hat zwei Vorteile:

1. **Parallelisierbarkeit.** Alle DQN-Agenten können parallel trainiert werden.
2. **Stabilität.** Der Lernvorgang wird dadurch stabilisiert. Der Agent startet ohne Vorwissen mit randomisierten Aktionen. Wenn an allen Maschinengruppen mit randomisierten Aktionen gestartet wird, bricht die Linienperformance zusammen. Ungleichmäßige Zulieferungsmuster an Maschinengruppen spiegeln keine normale Situation wider.

Beginnt der DQN-Agent ohne Vortraining in einer überwachten Lernphase, sollte der Anteil der explorativen Aktionen anfangs bei 100 % liegen. Das NN hat anfangs noch kein Modell über seine Umgebung gebildet, weshalb zielgerichtete Handlungen nicht damit ermittelt werden können. Während

des Lernvorgangs in **Phase A** wird der Anteil der explorativen Aktionen gleichmäßig abgesenkt (*linear annealed ϵ-greedy*), bis die Wahrscheinlichkeit für eine explorative Aktion bei etwa 10 % – 20 % liegt.

In der zweiten Lernphase, **Phase B**, wird mit allen DQN-Agenten gearbeitet. Allerdings lernt immer nur ein DQN-Agent. Dies ist notwendig, um das System stabil und konstant genug zu halten, damit der lernende Agent den Einfluss seiner Aktionen beobachten kann. Würden die anderen DQN-Agenten ihr Verhalten gleichzeitig verändern, könnte der aktive DQN-Agent nicht klar korrelieren, ob Veränderungen durch sein Verhalten oder das der anderen hervorgerufen wurden.

Daher befinden sich die DQN-Agenten während des aktiven Lernens eines anderen Agenten jeweils in einem inaktiven Modus. Trotzdem werden die Aktionen an der jeweiligen Maschinengruppe mit den NNs bestimmt, deren Gewichte jedoch unverändert bleiben. Dadurch kann sich der lernende DQN-Agent an das Verhalten der anderen Agenten anpassen.

Im Gegensatz zu **Phase A**, wo eine lokale Bewertung gegeben wurde, wird in **Phase B** mit einer globalen Bewertung gearbeitet. Die Bewertungsfunktion sollte im Idealfall stetig, aber mindestens näherungsweise stetig, von lokaler zu globaler Bewertung bei gleichbleibender *Policy* π übergehen. Falls dies nicht der Fall ist, werden die Vorteile der getrennten Lernphasen nicht genutzt und das System muss neu angelernt werden.

> **Beispiel: Übergang von lokaler zu globaler Bewertungsfunktion für *Loseigenschaften* und *Maschineneigenschaften***
>
> Im Folgenden wird anhand der Beispiele *Loseigenschaften* und *Maschineneigenschaften* erläutert, wie die Bewertungsfunktion näherungsweise stetig von lokaler zu globaler Bewertung übergeht. Auftragseigenschaften, wie beispielsweise die Abweichung vom geplanten Liefertermin, werden einzelnen Losen zugeordnet. Dann wird als lokale Funktion der Mittelwert der Auftragseigenschaften der Lose an der Maschinengruppe verwendet. Dieser sollte näherungsweise dem Mittelwert der Abweichung der geplanten Liefertermine aller Lose in der Linie entsprechen,

wenn das Paradigma gilt, dass eine gleichmäßige Abarbeitungsrate eines Auftrags erstrebenswert ist (Grundlage von ODD). In **Phase B** wird als globale Bewertung dann der Mittelwert der Loseigenschaft aller Lose in der Linie herangezogen.

Bei Maschineneigenschaften wird der Mittelwert der Maschineneigenschaft der Maschinen der Anlagengruppe verwendet, z.B. definiert durch die lokale Auslastung. Als globale Eigenschaft wird dann der Mittelwert der Maschineneigenschaft über alle Maschinen verwendet. In einer ausgetakteten Linie sollte die Auslastung an allen Maschinen in etwa identisch sein.

Der Anteil der explorativen Aktionen ist in **Phase B** konstant bei etwa 10 % – 30 %.

Die DQN-Agenten werden über eine Multi-Agenten-Umgebung gesteuert. Diese ordnet Ereignisse der jeweiligen DQN-Agenten-Umgebung zu und steuert, welcher DQN-Agent aktiv trainiert wird. In den einzelnen Umgebungen der Agenten sind nur die Ereignisse sichtbar, die sich auf die entsprechende Maschinengruppe beziehen. Dadurch besitzt jede Umgebung eine andere Zeitdiskretisierung. Die Multi-Agenten-Umgebung bildet die Vereinigung aller Zeitdiskretisierungen der einzelnen Umgebungen.

Die Aufteilung der Agenten während des Trainings ist in Abbildung 6.5 gezeigt.

Der Lernvorgang kann prinzipiell direkt in **Phase B** gestartet werden. Abhängig von der Art der Fabrik und der Anzahl an Agenten ist jedoch ein Vielfaches der Rechenzeit zu erwarten.

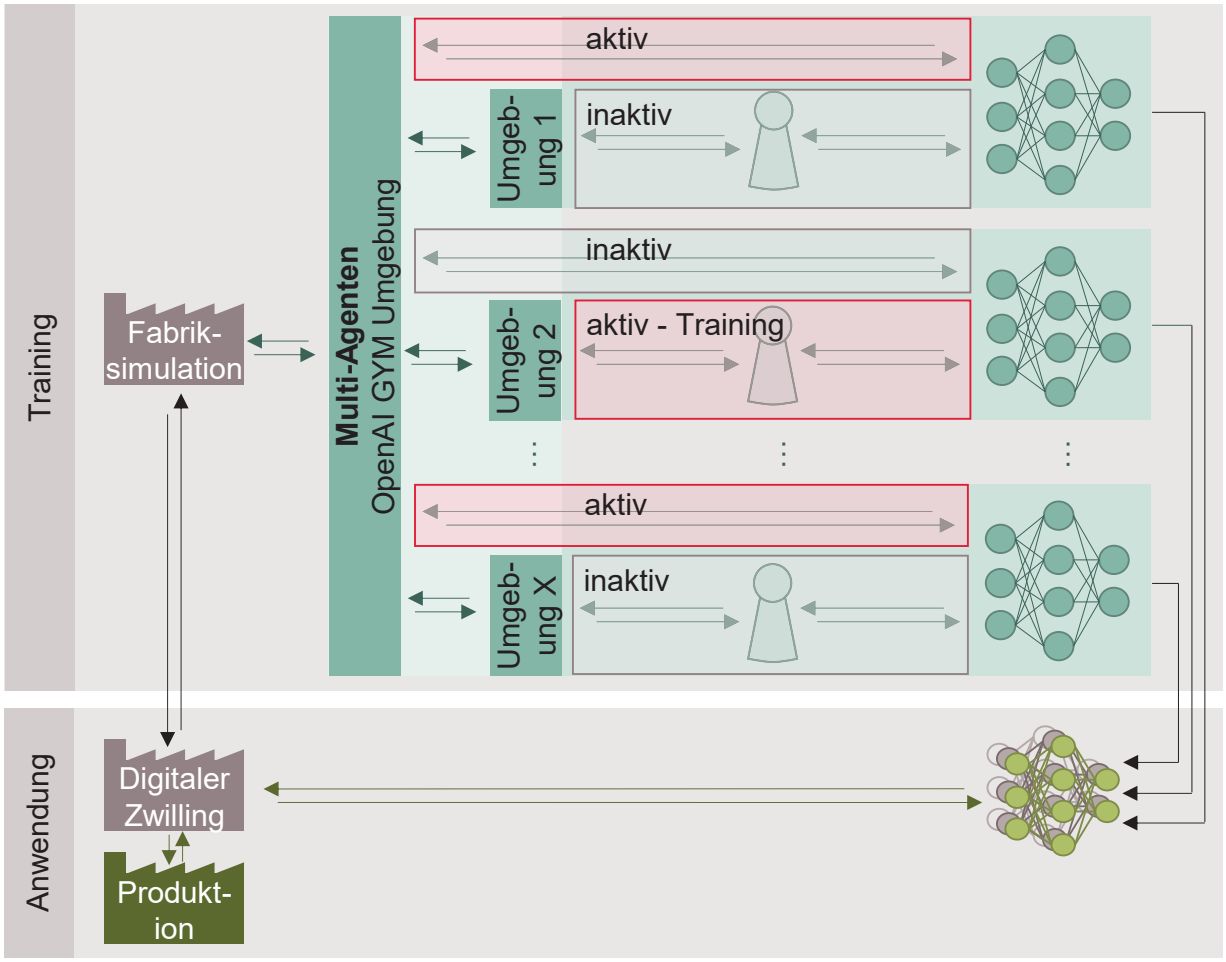


Abbildung 6.5.: Anlernen der DQN-Agenten im MAS und Übertragung der NNs in die Produktion

6.5. Einsatz des Systems in der Produktion

In die Produktion werden ausschließlich die NNs übertragen, die in der Produktion nicht mehr verändert werden. Da die NNs nicht mehr trainiert werden, sind keine explorativen Aktionen notwendig. Die konstanten NNs erzeugen kein neues unerwartetes Verhalten und garantieren die Stabilität der Produktion.

Die NNs in der Produktion können regelmäßig durch neu trainierte NNs ausgetauscht werden. Hierfür müssen nur die Gewichte, also die Parametrisierung, der NNs in der Produktion ersetzt werden. Eine Änderung an der Software-Implementierung ist nicht notwendig.

Bei diversen Veränderungen in den KPIs oder dem Portfolio der Produktion können die NNs schnell und flexibel neu trainiert werden. Der einfache Austausch der NNs in der Produktion garantiert eine hohe Flexibilität. Die Trennung von Training und Anwendung hat zwei Vorteile:

- **Leistungsfähigkeit der Produktionssteuerung während des Trainings.** Während des Trainings werden bis zu 100 % explorative Schritte ausgeführt. Dieses „Kennenlernen“ der Umgebung durch den DQN-Agent ist notwendig, um später effiziente Strategien abzuleiten. Dies führt zu einer stark verminderten Leistungsfähigkeit des Systems in der Trainingsphase, da die Aktionen zufällig ausgewählt werden. In die Produktion wird dann nur die „Quintessenz“ bzw. das Ergebnis des Lernprozesses übertragen.
- **Rechenzeit.** Während das Training von NNs rechenaufwendig ist, können Bestimmungen mit einem NN relativ schnell vorgenommen werden. Das aufwendige Training kann somit ausgelagert werden, während die sogenannte Inferenz der NNs direkt in der Produktion ausgeführt werden kann. Dies garantiert die Echtzeitfähigkeit des Systems.

> **Beispiel: Trainingszeiten aus den Fallbeispielen aus Kapitel 8**
>
> In den Fallbeispielen aus Kapitel 8 dauerte das Training auf dem in Kapitel 7.3 beschriebenen Rechner 1 — 2 Tage. Die Berechnung eines Inferenzschrittes, also eine Entscheidung über das nächste zu prozessierende Los, lag im Bereich von Millisekunden.

Voraussetzung für eine erfolgreiche Anwendung in der Produktion ist die realistische Modellierung der Fabrik in den entscheidenden Aspekten. Darauf wird näher bei der Implementierung der Methode in Abschnitt 7.2 eingegangen.

Wenn das Simulationsmodell und das physische System in den entscheidenden Aspekten nicht vollständig übereinstimmen, ist es möglich, die DQN-Agenten in der produktiven Anwendung beizubehalten. Die DQN-Agenten können dann in der Produktion mit einer verringerten Lernrate und

einem geringen Anteil explorativer Aktionen weiterlernen. Dadurch können theoretisch Unterschiede zwischen Simulationsmodell und physischem Produktionssystem überwunden werden. Allerdings senken die explorativen Aktionen in der Produktion die Leistung der Steuerung. Ohne oder mit einem geringen Anteil an explorativen Aktionen verändert der DQN-Agent sein Verhaltensmuster nicht mehr grundlegend. Das Vorgehen empfiehlt sich als Feinanpassung bei geringfügigen Unterschieden zwischen Produktion und Simulation.

Kapitel 7

Prototypische Implementierung der Methodik in einer Halbleiterscheibenfertigung

Eine Halbleiterscheibenfertigung mit einem diversifizierten Produktionsportfolio im *More-than-Moore*-Bereich ist ein typisches Beispiel für eine komplexe Werkstattfertigung. Im Folgenden wird erst eine allgemeine prototypische Implementierung beschrieben, um dann auf die Integration in eine bestehende Produktionssteuerung in einer Halbleiterscheibenfertigung, einem *Frontend*, einzugehen.

Bei der Implementierung des autonomen *Dispatchings* sind mehrere Herausforderungen zu beachten: Das Simulationsmodell muss als Grundlage für das RL den Qualitätsansprüchen genügen. In Kapitel 7.1 wird das eingesetzte Simulationsmodell beschrieben und in Kapitel 7.2 wird eine Methode

beschrieben, wie die Synchronisation zwischen physikalischer und digitaler Welt sichergestellt werden kann. Eine weitere Herausforderung ist die effiziente Implementierung des ML, auf die in Kapitel 7.3 eingegangen wird. Im nächsten Kapitel 7.4 wird aufgezeigt, wie das ML weiter beschleunigt werden kann. Danach wird die Integration in eine bestehende Fertigungssteuerung und ein MES beschrieben. Hierauf wird im letzten Teil dieses Kapitels, Abschnitt 7.5, eingegangen.

7.1. Aufbau und Parametrisierung des Simulationsmodells

Um eine möglichst hohe Flexibilität bei der Integration der Techniken des ML sicherzustellen, wurde das Simulationsmodell von Grund auf implementiert. Im Folgenden wird der Aufbau und Ablauf des Simulationsmodells erläutert.

Bei der Implementierung wurde auf zwei Paradigmen hohen Wert gelegt: Erstens, eine klare Modularisierung und Strukturierung durch objektorientierte Programmierung. Dadurch wird einerseits sichergestellt, dass das Simulationsmodell langfristig gepflegt werden kann. Andererseits macht es den Austausch von Teilen des Modells einfach möglich und definiert klare Schnittstellen. Diese Funktionalität wird bei dem Vergleich verschiedener Steuerungen verwendet.

Das zweite Paradigma ist die klare zeitliche Strukturierung durch die Verwendung einer ereignisorientierten Simulation (engl. *Discrete Event Simulation (DES)*). Die sequenzielle Natur der ereignisorientierten Simulation vereinfacht die Überführung in einen Markow-Entscheidungsprozess, der für das RL notwendig ist.

Das Simulationsmodell baut auf fünf Klassen auf, die in Abbildung 7.1 gezeigt sind und im Folgenden beschrieben werden. Der Aufbau der Produktionslinie und alle notwendigen Parameter der Simulation werden in der Klasse *modelParameter* festgelegt. Sie dient als Datenspeicher und zur Validierung aller Eingaben. Mit den Parametern aus *modelParameter* wird in der Klasse *model* dann das eigentliche Modell aufgebaut. Während in

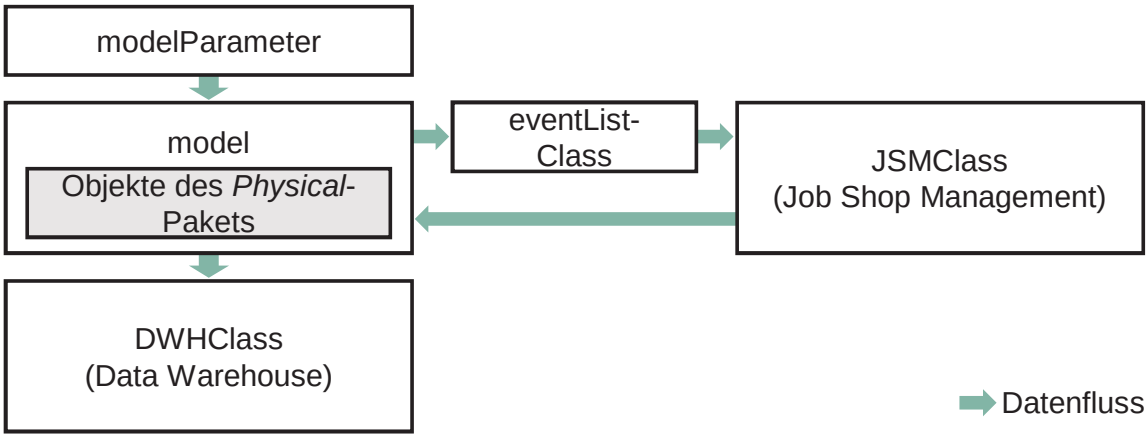


Abbildung 7.1.: Datenfluss zwischen den Hauptklassen der Fabriksimulation

der Klasse *modelParameter* beispielsweise nur die Anzahl der Maschinen eines Typs festgelegt sind, erzeugt die Klasse *model* für jede Maschine ein eigenes Objekt. Alle Objekte bis auf *DWHClass (Data Warehouse Class)*, die von der Klasse *model* erzeugt werden, haben in der Fabrik einen physischen Gegenpart (Maschinen, Lose, Transportsystem, …). Die Klassen dieser Objekte sind im *P (Physical)*-Paket zusammengefasst, das noch ausführlicher beschrieben wird.

Die Klasse *DWHClass*, die *Data Warehouse Class*, nimmt eine Sonderstellung ein, da das Objekt dieser Klasse von *model* erzeugt wird, aber in der Produktion keine physische, sondern nur eine virtuelle Repräsentation besitzt. Alle anderen Objekte berichten an dieses Objekt. Dies dient der Auswertung und Dokumentation der Simulation.

Alle Objekte aus Klassen des *P*-Pakets können Ereignisse erzeugen, die sie bei der Instanz von *eventListClass* registrieren. Die Klasse *eventListClass* sortiert und filtert die Ereignisse.

Das aktuelle Ereignis wird über die Funktion *getNextEvent()* an das *Job Shop Management (JSM)* übergeben, das ein Objekt der Klasse *JSMClass* ist. Primär definiert die Klasse ein *Interface* einer Produktionssteuerung mit der restlichen Simulation. Dem *JSM* wird das aktuelle Event übergeben. Das *JSM* reagiert dann auf das Event und führt eine Aktion in der Simulation

Algorithmus 7.1 Ablauf der Simulation

```
1: mP = modelParameter()
2: eL = eventListClass()
3: m = model(mP, eL)
4: JSM = JSMClass(m, eL)
5: While eL.isNotEmpty()
6:       JSM.processEvent(eL.getNextEvent())
7: End While
```

aus. Diese Aktion kann ein weiteres Event registrieren. Ein Beispiel für eine Aktion wäre das Starten eines Loses auf einer Maschine. Auf der Maschine wird dann der Zufallsanteil der Prozesszeit ermittelt und das *MOVEOUT* Event wird dann für die berechnete Zeit in der Event-Liste registriert. Der Algorithmus ist in Algorithmus 7.1 gezeigt und das Zusammenspiel der Klassen ist in Abbildung 7.1 dargestellt.

Die *JSMClass* definiert ein *Interface*, das von mehreren Steuerungsmechanismen genutzt werden kann. Im Folgenden wird dies verwendet, um die Simulation mit unterschiedlichen Produktionssteuerungen laufen zu lassen.

Die Klassen des *P*-Pakets sind in Abbildung 7.2 aufgeführt. Zwei Hauptklassen werden unterschieden: *Lot* und *QueueClass*. Alle relevanten Elemente in der Fabrik sind entweder selbst ein Auftrag (*Lot*) oder sie können ein Los (*QueueClass*) aufnehmen. Von *QueueClass* erben drei Klassen: *Transport*, *Workcenter* und *Equipment*. Die Klasse *Transport* stellt ein Transportsystem dar. Die Klasse *Workcenter* stellt ein lokales Lager zur Verfügung und bündelt Maschinen der Klasse *Equipment*. Damit wird in der Steuerung zuerst entschieden, an welche Maschinengruppe geliefert wird, und dann kann vom Zwischenlager entschieden werden, welche Maschine des *Workcenters* den Auftrag bearbeitet.

Die Klasse *Equipment* stellt eine Maschine dar. Anhand der Parameter der Klasse können damit beliebige Maschinentypen erstellt werden. Von der Klasse *Equipment* erben die zwei Klassen *ELoad* und *EClosing*. Ein Objekt der Klasse *ELoad* „lädt“ neue Aufträge in die Produktion. Ein Objekt der Klasse *EClosing* schließt Aufträge in der Produktion ab. In der Halbleiterfertigung

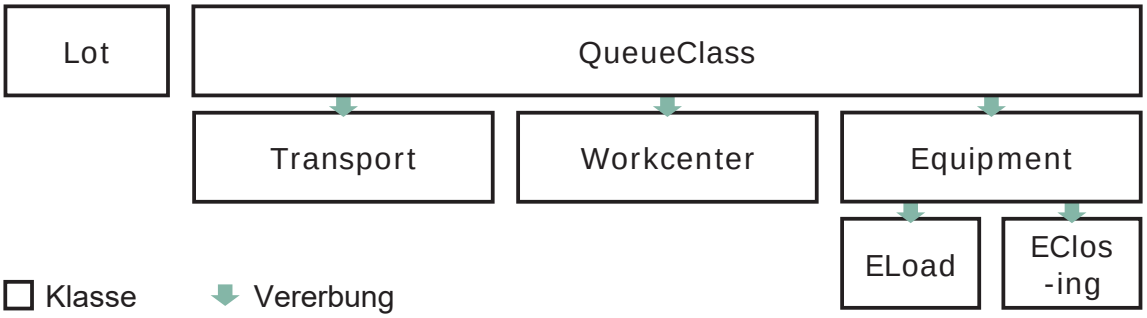


Abbildung 7.2.: Klassen des *P*-Pakets der Fabriksimulation. Die Klassen repräsentieren physische Objekte in der Produktion.

handelt es sich hierbei um die Ein- und Ausschleusung von Losen aus dem Reinraum.

Alle abrufbaren Parameter der Klassen des *P*-Pakets werden dem äquivalenten Objekt in der Produktion dort zugeordnet. Es handelt sich dabei also nicht um Metaparameter, die für die Simulation notwendig sind, sondern um physikalische und logistische Größen der Objekte.

Die Simulation wurde in MathWorks® MATLAB in der Version R2015b implementiert (MATLAB, 2015).

Die Simulationsumgebung wurde für die Experimente in Kapitel 8 eingesetzt. Es handelt sich dabei nicht um einen Digitalen Zwilling. Anfangs wurde eine automatisierte Parametrisierung implementiert, die die *modelParameter*-Datei aufbauend auf den produktiven *Data-Warehouse*-SQL-Datenbanken der Infineon Technologies AG erstellt. Dabei wurden die Arbeitspläne, die Auftragsfreigabe und Prozesszeiten aus den historischen Daten gewonnen. Aus drei Gründen wurde dieses Verfahren nicht weiter verfolgt:

- **Veränderungen.** In der Produktion werden laufend Verbesserungen umgesetzt, weshalb Statistik auf historischen Daten nur eingeschränkt möglich ist.
- **Datenqualität und -verfügbarkeit.** Die Datenqualität und -verfügbarkeit in den *Data-Warehouse*-Datenbanken erlaubte kein realistisches Abbild der Produktion.

- **Rechenzeit für ML.** Für diese Arbeit stand nicht ausreichend Rechenkapazität zur Verfügung, um ML für eine realistische Produktionsgröße umzusetzen.

In den Experimenten in Kapitel 8 wurden verkleinerte Linien in der Simulationsumgebung aufgebaut, die in den entscheidenden Parametern einer Halbleiterfertigung nachempfunden sind. Die verkleinerten Linien in den Fallbeispielen sind Ausschnitten aus der realen Produktion ähnlich. Bei der Parametrisierung wurde auf die realen Produktions- und Grunddaten zurückgegriffen, die aus den produktiven *Data-Warehouse*-SQL-Datenbanken der Infineon Technologies AG extrahiert wurden. So entsprechen beispielsweise Prozesszeiten und Arbeitspläne der Technologieklassen den realen Pendants.

Allerdings nimmt die Simulation einige Vereinfachungen vor, die eine komplette Parametrisierung durch reale Daten verhindern. In die Simulation gehen beispielsweise keine menschlichen Eigenschaften der Werker und keine *ENGINEERING TIME* der Prozessentwicklung ein (siehe Abbildung 2.4). Zudem sollte in den Fallbeispielen ein klarer wiedereintretender Fluss modelliert werden, der in der realen Produktion erst nach längeren Prozessfolgen auftritt. Diese Veränderungen führten dazu, dass logistische Kerngrößen nicht im erwarteten Bereich lagen. Daher wurden die Parameter der Simulation behutsam angepasst, um eine Betriebskennlinie und einen dazugehörigen Betriebspunkt zu erreichen, die einer realen Halbleiterfertigung entsprechen (siehe Kapitel 2.4). Stimmen die logistischen Kerngrößen in der Werkstattfertigung mit der Realität überein, sind die Ergebnisse in der Simulation prinzipiell übertragbar.

7.2. Qualität des Simulationsmodells

Im RL-Modell wird das *Dispatching* im Simulationsmodell optimiert. Für die Übertragung in die Produktion ist es daher entscheidend, dass Simulation und physische Produktion in den ausschlaggebenden, meist logistischen Aspekten übereinstimmen. Kleinere Unterschiede zwischen Simulation und

physischer Fabrik können ausgeglichen werden, indem in der Produktion mit geringer Lernrate weitergelernt wird (siehe Kapitel 6.5). Trotzdem sollte das Simulationsmodell so aktuell und realitätsnah wie möglich sein.

Die Qualität des Simulationsmodells kann durch die Verwendung eines Digitalen Zwillings und der Ankerpunktmethode sichergestellt werden (Talkhestani et al., 2018). Ein Digitaler Zwilling ist ein integriertes Simulationsmodell eines Systems, das die Parameter, Eigenschaften und Prozesse des physischen System in der virtuellen Welt in Echtzeit abbildet (Talkhestani et al., 2018). Damit wird der Zustand des physischen Objekts in die virtuelle Welt gespiegelt. Die Verknüpfung eines Digitalen Zwillings mit einem physischen Objekt ist Kern von cyber-physikalischen Systemen.

Mit der Ankerpunktmethode wird die Synchronisierung von digitalem und realem Objekt sichergestellt (Talkhestani et al., 2017; Talkhestani et al., 2018). Ein Ankerpunkt wird definiert als Daten oder Information einer mechatronischen Komponente, die genutzt werden können, um Veränderungen im physischen System zu detektieren (Talkhestani et al., 2018). Dabei hängen Ankerpunkte in den Domänen Mechanik, Elektronik und Software zusammen und sind miteinander verknüpft (Talkhestani et al., 2018).

Ankerpunkte in Software und Elektronik können größtenteils automatisiert detektiert werden (Talkhestani et al., 2017). Änderungen in der Mechanik müssen meistens von Ingenieuren als Änderungen dokumentiert werden (Talkhestani et al., 2017). Durch eine Konsistenzprüfung der Domänen untereinander können mechanische Änderungen auch über Software und Elektronik erkannt werden.

Die Ankerpunkte werden in einem wissensbasierten Änderungsmanagementsystem gesammelt (Talkhestani et al., 2017). Dort werden sie einer regelbasierten Konsistenzprüfung unterzogen, wobei Konsistenz als Freiheit von Widersprüchen verstanden wird (Talkhestani et al., 2017). Dabei wird auf Konsistenz in Datenflüssen und Prozessen sowie Konsistenz zwischen den Domänen geachtet.

Mit den Ergebnissen des Änderungsmanagement-System kann der Digitale Zwilling aktualisiert werden bzw. auf Inkonsistenzen hingewiesen werden. Das vollständige Modell ist in Abbildung 7.3 gezeigt. Mehrere Beispiele

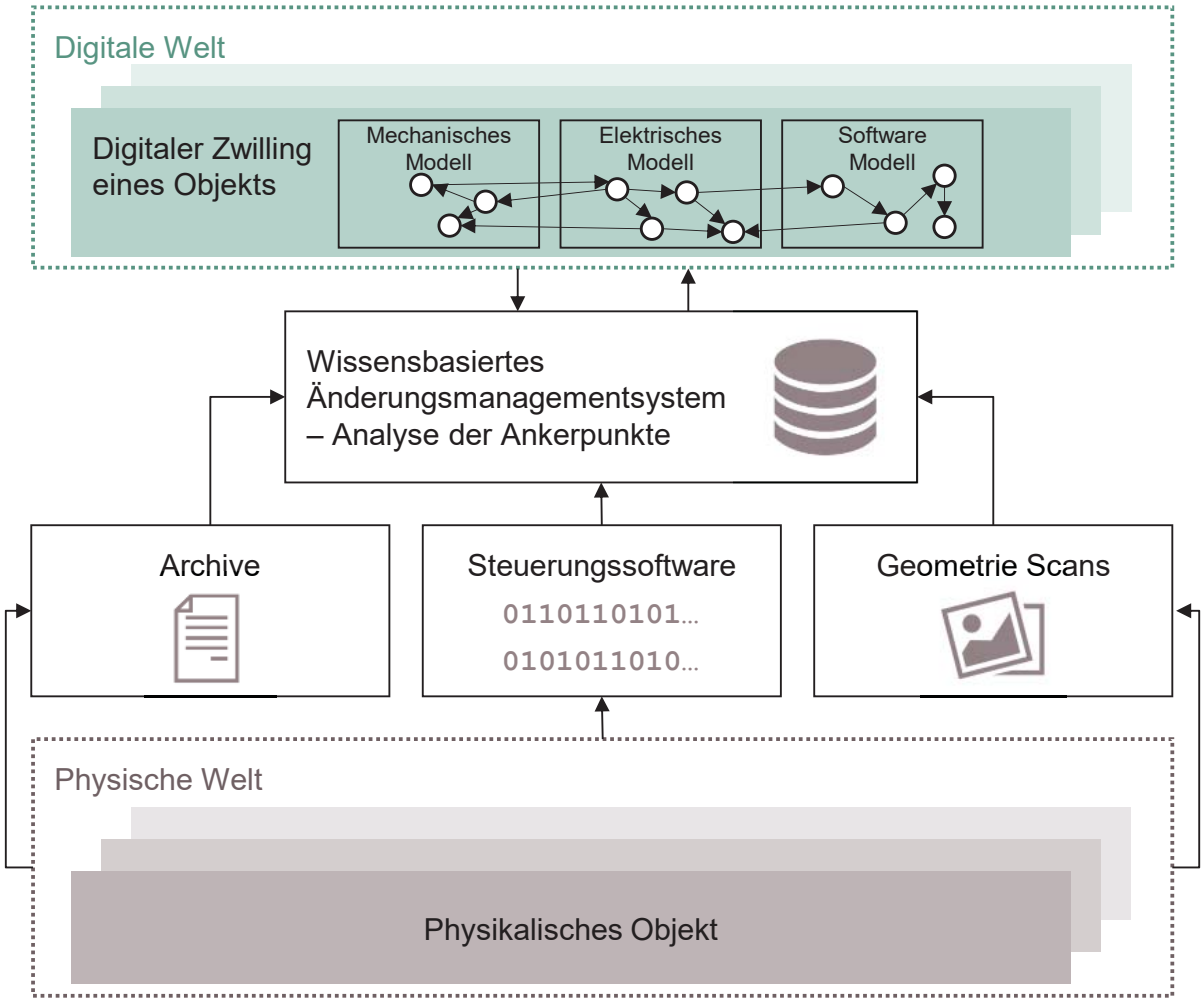


Abbildung 7.3.: Über ein wissensbasiertes Änderungsmanagementsystem wird der Digitale Zwilling mit dem physikalischen Objekt synchronisiert. Dies geschieht über eine Konsistenzprüfung der erfassten Ankerpunkte. Abbildung basiert auf Talkhestani et al. (2018)

für die Synchronisation von einem Digitalem Zwilling mit der Produktion werden von Rosen et al. (2015) beschrieben.

Das Simulationsmodell des Digitalen Zwillings schafft damit optimale Voraussetzungen für das autonome *Dispatching*. Die Übertragbarkeit der Ergebnisse des autonomen *Dispatching* in die physische Produktion wird durch den Digitalen Zwilling garantiert.

7.3. Implementierung des Maschinellen Lernens

Die populärste Programmiersprache für ML ist *Python* (Stand Oktober 2016, siehe Puget, 2016). Da die benötigten Bibliotheken nicht in MathWorks® MATLAB verfügbar sind, wird für die Implementierung des MLs *Python* eingesetzt. Voraussetzung dafür ist die „MATLAB *API for Python*“ die es erlaubt, aus *Python* heraus eine MATLAB *Session* zu starten.

Die MATLAB *Session* wird in der Multi-Agenten-*OpenAI GYM*-Umgebung gestartet. *OpenAI GYM* bietet eine standardisierte Schnittstelle zwischen Agent und Umgebung (Brockman et al., 2016). Auf der Multi-Agenten-*OpenAI GYM*-Umgebung bauen dann mehrere separate *OpenAI GYM*-Umgebungen auf, die jeweils für einen Agenten die Schnittstelle darstellen. Die Multi-Agenten-*OpenAI GYM*-Umgebung koordiniert, an welche Agenten-Umgebung ein Ereignis in der Simulation weitergeleitet wird.

Als RL-*Framework* wird *keras-rl* verwendet (Plappert, 2016). Das *keras-rl-Framework* verwendet die *OpenAI GYM*-Schnittstelle für Umgebungen, besitzt eine Implementierung des DQN-Agenten und verwendet als Programmierschnittstelle (engl. *Application Programming Interface (API)*) für NNs *keras* (Chollet et al., 2015; Chollet, 2017). Die Programmierschnittstelle *keras* zeichnet sich durch eine hohe Benutzerfreundlichkeit und Flexibilität aus. Sie kann mit den NN-Bibliotheken *TensorFlow*, *CNTK* und *Theano* verwendet werden. In dieser Arbeit wird *TensorFlow* (Abadi et al., 2016) eingesetzt.

Für die beschleunigte Ausführung der NNs wird die Grafikkarte *GeForce GTX 1050Ti* des Herstellers Nvidia Corporation® verwendet. Die *CUDA API* von Nvidia® erlaubt ein Ausführen von universellem Programmcode auf der *Graphics Processing Unit (GPU)*, womit dort die Berechnungen für NNs durchgeführt werden können. Ansonsten wird ein handelsüblicher Rechner mit einer Intel® *Core*™ i7 — 4770 *Central Processing Unit (CPU)* mit 3,40 GHz eingesetzt. Für größere Fallbeispiele als in Kapitel 8 oder eine Anwendung in der Produktion ist ein leistungsstärkeres System empfehlenswert.

7.4. Beschleunigung des Maschinellen Lernens

Das ML, insbesondere das DL, ist enorm rechenaufwendig (LeCun et al., 2015). Für einen Einsatz des autonomen *Dispatchings* in der Produktion muss das Anlernen des Systems in vertretbarer Zeit von wenigen Tagen möglich sein. Daher ist ein sparsamer Umgang mit den *Hardware*-Ressourcen wichtig. Im Folgenden werden generelle und modellspezifische Methoden vorgestellt, wie das Training beschleunigt werden kann.

Eine zentrale Rolle dabei spielt die eingesetzte *Hardware*. Grafikkarten des Herstellers Nvidia Corporation® beschleunigen den Lernvorgang um das 10—20 fache im Vergleich zu CPUs (LeCun et al., 2015). Im besten Fall wird ein GPU-*Cluster* eingesetzt, in dem mehrere Grafikkarten zu einem Verbund zusammengeschlossen sind (Kindratenko et al., 2009). Spezialisierte Chips für DL sind heute teilweise bereits in der *Cloud* verfügbar wie beispielsweise Google LLCs *Tensor Processing Unit* (TPU; Jouppi et al., 2017; Stand August 2018). Der *Hardware*-Markt entwickelt sich rapide weiter und es ist sinnvoll, auf aktuelle Plattformen und Produkte zu setzen. Sicherheitsbedenken bei *Cloud*-Lösungen sollten allerdings nicht außer Acht gelassen werden.

Das Anlernen der NNs kann nicht nur durch spezifische *Hardware* beschleunigt werden, sondern auch durch *Software*. Hierbei können die Charakteristiken der Methode und das menschliche Fachwissen ausgenutzt werden, um den Lernvorgang abzukürzen. Beim autonomen *Dispatching* sind mehrere Verfahren in den Lernprozess integriert, die diesen beschleunigen:

- **Vortraining mit überwachtem Lernen.** Beim Vortraining mit SL wird das menschliche Vorwissen in einer bereits vorhandenen Steuerung genutzt, um die NNs vorzuprägen. Dies wurde genauer in Kapitel 6.3 beschrieben.
- **Multi-Agenten-System.** Einzelne DQN-Agenten im MAS können, wie in Kapitel 6.4 beschrieben, parallel trainiert werden.
- **Verwendung von konventionellen Steuerungen an passiven Maschinengruppen in Phase A.** Anfangs kann nur der lernende DQN-Agent seine Maschinengruppe mit einem NN steuern, während die

weiteren Maschinengruppen durch konventionelle Systeme gesteuert werden.

- **Aufteilung des Lernvorgangs mit lokaler und globaler Bewertung (Phase A und B).** Die DQN-Agenten lernen anfangs mit einer lokalen Bewertung schneller, da die Korrelation zwischen Aktionen und Bewertung direkter ist. Dies ist genauer in Kapitel 6.4 dargelegt.

Die vorgestellten Verfahren erlauben es, den Lernvorgang zu beschleunigen. Allerdings bergen sie die Gefahr, dass vorhandene *Dispatching*-Strategien zu stark in die NNs eingeprägt werden. Dadurch kann es passieren, dass neue Strategien nicht gefunden werden.

7.5. Integration in die bestehende Produktionssteuerung einer Halbleiterfertigung

Mit dem größten Marktanteil von 27 % nimmt Applied Materials, Inc. (AMAT) eine vorherrschende Stellung bei Produktionsanlagen für Halbleiterchips ein (Castellano, 2018). AMAT liefert sowohl die Produktionsanlagen als auch diverse Software für den Fabrikbetrieb, in dem die Anlagenintegration direkt gewährleistet ist.

Ein Standard in der Produktionssteuerung von Halbleiterfabriken ist der AMAT RTD (Applied Materials, Inc., 2013; Ham et al., 2015). Der RTD ist Teil von Applied Materials *Advanced Productivity Family (APF)* und wird daher teilweise auch als APF RTD® bezeichnet. Im Folgenden bezieht sich die Abkürzung RTD immer auf das Produkt von Applied Materials, den APF RTD®.

Laut dem Hersteller AMAT konnten durch den Einsatz des RTD der Wafer-Durchsatz um 10 % – 15 % und die Auslastung um 10 % gesteigert werden (Applied Materials, Inc., 2013).

Aufgrund der tiefen Integration des RTD in den Fabriken sollte dieser weiterhin genutzt werden. Der RTD garantiert den stabilen Betrieb mit einer Rückfallroutine bei Fehlern und die Integration und Schnittstellen zu Anlagen und MES. Neue Systeme lassen sich als Makros und zusätzliche

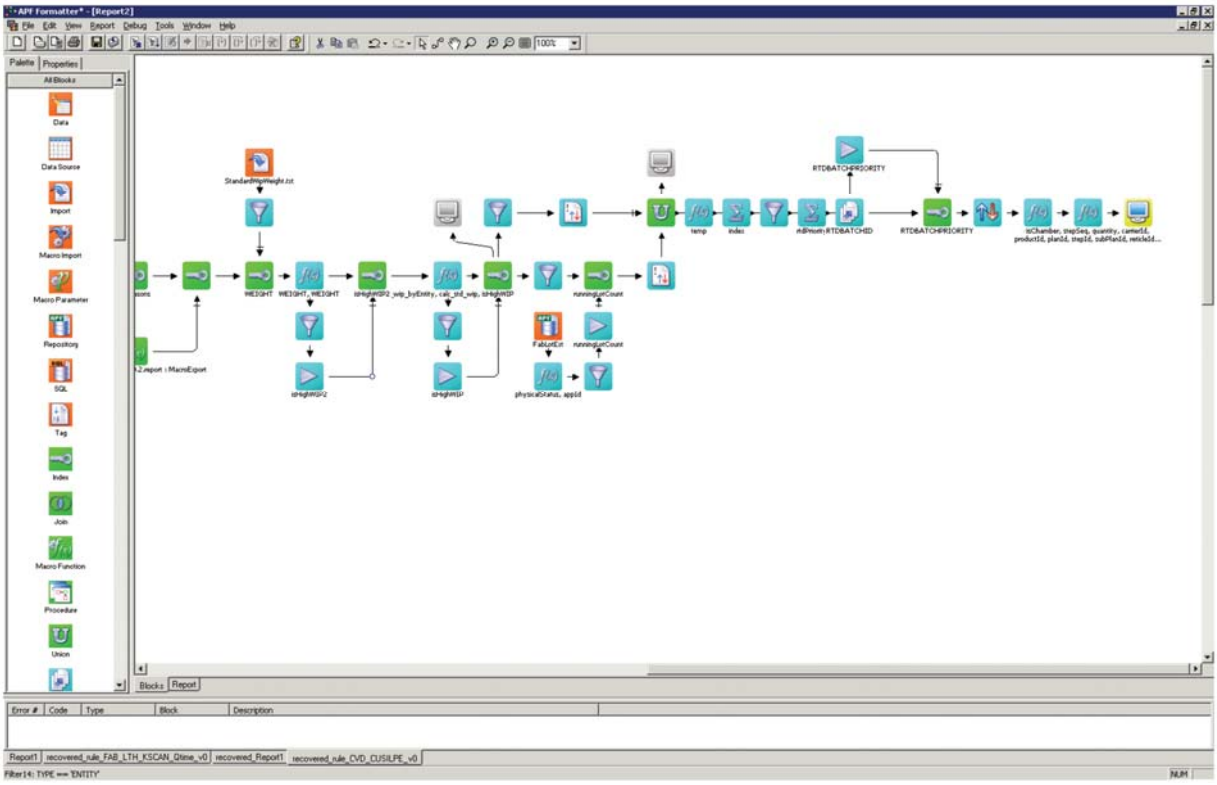


Abbildung 7.4.: Bildschirmfoto des Applied Materials, Inc. APF *Formatters*. Mit dem APF *Formatter* werden die RTD-Regeln erstellt. Er enthält Kacheln für den Aufruf externer Programme sowie für das Laden von Daten aus externen Datenquellen, beispielsweise SQL-Datenbanken. Quelle: Applied Materials, Inc., 2013

Datenquellen an den RTD anschließen. Diese Möglichkeit wurde in der Infineon Technologies AG bereits für die Anbindung eines mathematischen Optimierers an das *Dispatching*-System genutzt.

Im RTD werden Regeln im APF *Formatter* aufgebaut. Wie in Abbildung 7.4 gezeigt, werden dort Kacheln mit unterschiedlichen Funktionen zu Regeln verbunden. Beim RTD handelt es sich damit im weiteren Sinne um eine *Workflow Engine* mit spezifischen Modulen bzw. Kacheln für die Produktionssteuerung in der Halbleiterfertigung. Mit den Kacheln lassen sich Hierarchien unterschiedlicher Heuristiken aufbauen.

Eine Produktionssteuerung mit NNs würde als externes Programm vom RTD aufgerufen werden und die Prioritätsbewertungen der Aufträge dem

RTD als externe Datenquelle bereitstellen. Bei einem externen Aufruf werden zuerst relevante Daten aus dem RTD *Repository*, der RTD-Datenquelle, extrahiert und in eine Datenbank oder eine Textdatei exportiert. Über ein Makro wird dann das externe Programm aufgerufen, das auf den RTD-Datenexport zugreift. Nach der Berechnung im externen Programm speichert die externe Software ihre Daten in einer SQL-Datenbank. Diese wird vom RTD wiederum über eine Kachel eingelesen. Der RTD integriert die Daten dann in das RTD *Repository*. Somit können beispielsweise externe Auftragsprioritäten geladen und dann exekutiert werden.

Die bestehenden Heuristiken im RTD können beibehalten werden. Sie werden genutzt, wenn der Aufruf einer externen Software nicht erfolgreich war, bzw. aus der externen Datenquelle keine Daten geladen werden können. Somit wird eine neue innovative Methode mit einer stabilen, etablierten Lösung kombiniert und die Vorteile beider Methoden werden genutzt.

7.6. Interpretierbarkeit der Ergebnisfindung

NNs werden häufig als *Black-Box*-Modelle bezeichnet: Es ist für Menschen nicht ersichtlich, wie sie eine Klassifizierung vornehmen (Guidotti et al., 2018). Diese Eigenschaft der NNs führt zu Misstrauen in den Ansatz (Lipton, 2018). In einer Produktion, deren oberstes Ziel die Stabilität ist, existiert eine gewisse Zurückhaltung, produktionskritische Prozesse und Entscheidungen von NNs abhängig zu machen.

Eine entscheidende Rolle spielt dabei das Vertrauen in die Methode (Lipton, 2018; Ribeiro et al., 2016). Die Ergebnisse eines Optimierers sind ebenfalls nicht einfach nachvollziehbar, da keine hierarchische Entscheidungsfindung vorliegt. Allerdings herrscht hier ein hohes Vertrauen in die Methode, weshalb sie in der Halbleiterfertigung weit verbreitet ist (Ham et al., 2015).

Für NNs existieren diverse Methoden, um die *Black-Box*-Modelle zu visualisieren oder die Interpretierbarkeit zu erhöhen. Im Fallbeispiel in Kapitel 8 konnten durch Beobachtung der Aktionen der DQN-Agenten durch Experten bereits Rückschlüsse auf die Entscheidungslogik des NNs gezogen werden.

Daher wurde in dieser Arbeit keine Methode angewandt, die die Interpretierbarkeit des NNs erhöht. Trotzdem soll im Folgenden kurz auf Möglichkeiten eingegangen werden, die die Interpretierbarkeit von und schlussendlich das Vertrauen in NNs steigern können.

Eine Methode setzt an den Eingangsdaten an: Bei *Saliency Masks* wird systematisch bewertet, welche Eingangsinformation das NN für seine Klassifikation verwendet (Guidotti et al., 2018). Dabei werden Teile der Eingangsdaten abgedeckt und die Auswirkung auf die Klassifikation des NN beobachtet.

Bei Sensitivitätsanalysen wird die Korrelation der Unsicherheit bzw. Variabilität zwischen Eingangs- und Ausgangsdaten betrachtet (Guidotti et al., 2018).

Das *Local-Interpretable-Model-agnostic-Explanation (LIME)*-Verfahren setzt bei der Lokalität an (Ribeiro et al., 2016). Dabei wird eine lokale Menge an Erklärungen bereitgestellt und ein Optimierungsproblem formuliert, das Erklärungen zu lokalen Beispielen bzw. Datenpunkten korreliert.

Ein weiterer Ansatz ist das Überführen von NNs in Entscheidungsbäume oder Entscheidungsregeln (Guidotti et al., 2018). Zumindest lokal können diese Ansätze klare Entscheidungsregeln liefern. Frosst et al. (2017) vom Google *Brain Team* präsentieren beispielsweise eine Methode, wie ein NN in einen hierarchischen Entscheidungsbaum überführt werden kann.

Zahavy et al. (2016) präsentieren eine Methode spezifisch für die Analyse von DQN-Agenten. Dabei werden die Aktivierung der letzten verdeckten Ebene, die Information des Gradientenverfahrens sowie die Eingangsdaten aufgezeichnet. Auf die hochdimensionalen Aktivierungsdaten wird der *t-distributed-Stochastic-Neighbor-Embedding (t-SNE)*-Algorithmus (Maaten et al., 2008) zur Dimensionalitätsreduktion und Visualisierung angewendet. Das Ergebnis des t-SNE muss allerdings weiterhin manuell interpretiert werden.

Alle vorgestellten Ansätze oder Methoden haben Vor- und Nachteile. Alle liefern einen Beitrag zur Interpretierbarkeit von NNs und DQN-Agenten. Wenn das Vorgehen eines DQN-Agenten unklar ist, oder der Einsatz des autonomen *Dispatchings* am Vertrauen in die Methode scheitert, können die

aufgelisteten Erklärungsverfahren helfen, das *Black-Box*-Modell transparenter zu machen.

Wie bereits am Anfang des Kapitels erwähnt, wurden die Verfahren in dieser Arbeit nicht getestet. Wenn die zu optimierende Umgebung hinreichend bekannt ist, können Experten aus dem Verhalten des DQN-Agenten bereits weitreichende Rückschlüsse auf dessen Logik ziehen. Die aufgezählten Ansätze können als Ansatzpunkte für Anwender der Methode gesehen werden, für die die Entscheidungen des DQN-Agenten nicht nachvollziehbar sind.

Kapitel 8

Fallbeispiele und kritische Bewertung

Nach der Vorstellung der Methodik zur autonomen Produktionssteuerung und der prototypischen Implementierung wird die Methodik in diesem Kapitel auf den Prüfstand gestellt. Hierbei handelt es sich um eine Validierung, also Prüfung der festgelegten Anforderungen in Fallbeispielen, und nicht um eine Verifizierung (von lat. *veritas* „Wahrheit“ und *facere* „machen“), die eine absolute Wahrheitsaussage über die Methodik macht. Wie in Kapitel 1.4 im Begründungszusammenhang beschrieben, folgt die Arbeit dem Konzept der Falsifikation nach Popper (1973, S. 6 ff.; 2013, S. 12 ff.), das nur eine Validierung zulässt.

In den Beispielen wird die Methodik auf eine Produktionssteuerung zweier vereinfachter *Frontend-of-line*-Scheibenfertigungen angewandt. Im ersten Fallbeispiel wird die Auslastung optimiert. Im zweiten Fallbeispiel ist die Liefertreue die logistische Zielgröße der Produktionssteuerung. Zuletzt wird die Methodik anhand der Anforderungen aus Kapitel 5.1 kritisch reflektiert.

8.1. Fallbeispiel zur Auslastungsoptimierung[1]

Im Folgenden wird zuerst das zugrundeliegende Simulationsmodell vorgestellt. Danach wird die Wahl der Hyperparameter für das ML-Modell diskutiert und die Bewertungsfunktion formuliert. Zuletzt werden die Resultate des ersten Fallbeispiels präsentiert.

8.1.1. Simulierte Produktionslinie

Für das erste Experiment wird eine vereinfachte *Frontend-of-line*-Halbleiterfertigung modelliert (siehe Kapitel 4.1). Diese besteht aus vier unterschiedlichen Maschinengruppen. Die erste Maschinengruppe wird von zwei Lithographie-*Clustern* gebildet. Die *Cluster* führen die drei EPAs für das Belacken, Belichten und Entwickeln in einem vollautomatisierten Prozess aus. Da der Fokus des Experiments auf der Abarbeitungsreihenfolge der Aufträge liegt und diese während der drei EPAs nicht mehr verändert werden kann, werden die drei EPAs hier als ein Prozessschritt betrachtet. Für jede Technologieklasse liegt jeweils nur eine Fotomaske (engl. *Reticle*) vor. Daher können Aufträge der gleichen Technologieklasse nicht gleichzeitig auf beiden Maschinen prozessiert werden.

In der Maschinengruppe 2 benötigt die Maschine zur Ionenimplantation, im folgenden als *Implanter* bezeichnet, unterschiedliche Rüstungen für unterschiedliche Technologieklassen. Die Rüstzeiten sind in Tabelle 8.1 gegeben. *Implanter* haben einen hohen Durchsatz, weshalb nur eine Maschine benötigt wird.

Die nächsten EPAs sind in der Simulation in einem *Buffer*-Schritt zusammengefasst. Die Prozesse haben weniger Nebenbedingungen, erfordern kein besonderes *Dispatching* und haben einen geringeren Kapitaleinsatz, weshalb eher eine geringe Überkapazität bereitgestellt wird. In diesem Schritt wird als Nebenbedingung ein Transportbatching zu den Öfen in der Maschinengruppe 4 eingeführt. Dies kann der Fall sein, wenn Maschinen nicht an das automatische Transportsystem angeschlossen sind, sondern mit einem

[1] Das Kapitel basiert auf der Veröffentlichung Waschneck et al. (2018c).

(willkürliche Zeiteinheiten)	TK1	TK2	TK3
TK1	0,0	1,1	1,9
TK2	4,1	0,0	3,2
TK3	1,3	3,0	0,0

Tabelle 8.1.: Rüstzeiten für das Umrüsten von einer Technologieklasse (TK) zur anderen am *Implanter*

Transportwagen beliefert werden. Der Transportwagen wird vom Werker zur Maschinengruppe 4 gebracht, wenn er mit acht Aufträgen befüllt ist.

In der letzten Maschinengruppe in der Linie, Maschinengruppe 4, stehen drei Öfen. Die Öfen verarbeiten jeweils zwei Aufträge der gleichen Technologieklasse. Beim *Batching* kommt es nicht auf den Abarbeitungsgrad des Arbeitsplanes, sondern ausschließlich auf die Technologieklasse an. Außerdem gibt es eine Zeitkopplung mit dem vorhergehenden Prozess: Der Ofenprozess darf nicht mehr als 163 Minuten nach der EPA am *Buffer* gestartet werden. Eine Zeitkopplung zwischen *Implanter*, *Buffer* und Ofen ist ungewöhnlich. Sie ist normalerweise zwischen einem Reinigungsschritt und dem Aufwachsen des *Gate*-Dielektrikum des Transistors im Ofen notwendig (Pfeffer, 2012, S. 11). Dies ist in der Simulation nicht abgebildet. Da die Restriktion der Zeitkopplung trotzdem eingebracht werden sollte, wurde sie wie beim *Gate*-Dielektrikum vor den Ofenprozess gesetzt.

In der Simulation werden drei unterschiedliche Technologieklassen gefertigt. Aufbauend auf einer Technologieklasse können unterschiedliche Produkte realisiert werden, die sich insbesondere durch die eingesetzten Fotomasken unterscheiden. In diesem Experiment wird davon ausgegangen, dass die Masken im *Frontend-of-line* innerhalb einer Technologieklasse identisch sind. Die Produktspezialisierung findet dann im *Backend-of-line* statt, in dem die Transistoren durch die Metallisierungsebenen zu Schaltkreisen verbunden werden.

Die RPZs der EPAs sind, wie der Aufbau der Linie, die Nebenbedingungen und die *Dispatching*-Heuristiken, in Abbildung 8.1 angegeben. Die Zeiten

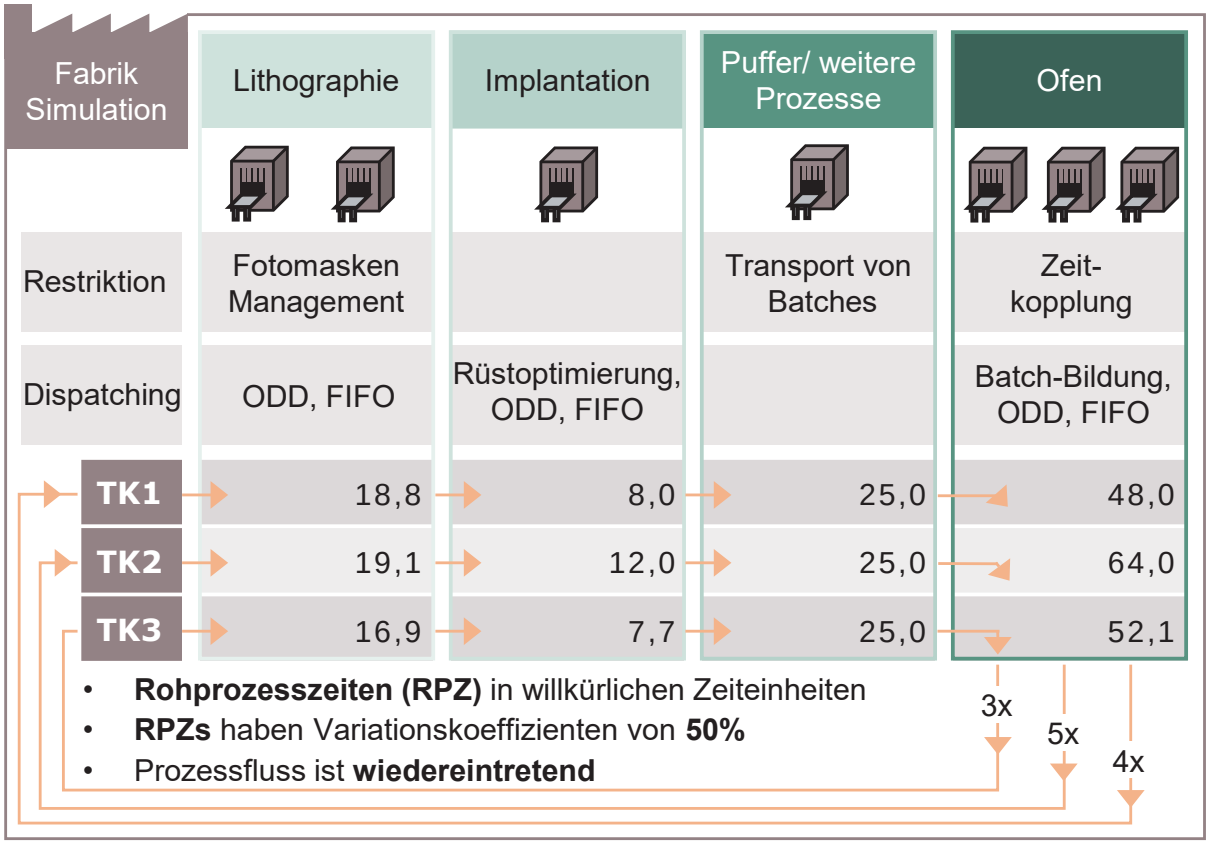


Abbildung 8.1.: Übersicht über das Simulationsmodell für Fallbeispiel 1 mit drei Technologieklassen (TKs) und vier Maschinengruppen. Die spezifischen Nebenbedingungen und *Dispatching*-Heuristiken der einzelnen Maschinengruppen werden aufgeführt. Für die Technologieklassen sind die RPZs gezeigt.

und Verteilungen entsprechen typischen Werten bei der in der Halbleiterfertigung[1]. Alle RPZs folgen einer Normalverteilung mit einem Variationskoeffizienten von 50 %. Der Variationskoeffizient von RPZs in der Halbleiterfertigung liegt zwischen $5-30$ %. In diesem Experiment wird ein deutlich höherer Variationskoeffizient von 50 % auf den RPZs verwendet, um damit implizit Maschinenausfälle und unterschiedliche Transportzeiten zu modellieren. Maschinenausfälle und Transportzeiten werden nicht explizit betrachtet.

[1] Die Werte wurden anhand von Daten der Infineon Technologies AG für Standardprozesse in der Halbleiterindustrie bestimmt.

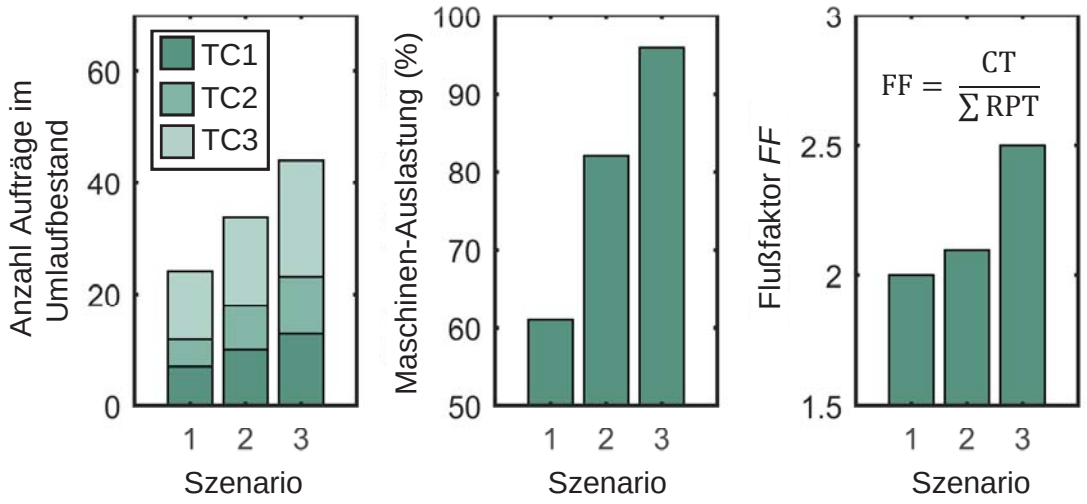


Abbildung 8.2.: Charakterisierung der drei Szenarien in der Fabriksimulation.

Alle Technologieklassen folgen einem wiedereintretenden Fluss. Dies bedeutet, dass die Linie von jedem Auftrag mehrfach durchlaufen wird. Die Technologieklasse TK1 durchläuft die Linie vier Mal, TK2 hat fünf Durchläufe, und TK3 zieht drei Schleifen. Der wiedereintretende Fluss kann ein Aufschaukeln der Linie erzeugen.

Im Experiment werden drei Szenarien mit unterschiedlichem Umlaufbestand betrachtet. Aus Umlaufbestand und vorhandener Kapazität wird der FF berechnet, der weitreichenden Aufschluss über das Management einer komplexen Werkstattfertigung gibt (siehe Kapitel 2.4). Ein niedriger FF, also zwischen $1,5-2,0$, bedeutet kurze Durchlaufzeiten, aber eine niedrigere Auslastung. Ein höherer Flussfaktor ermöglicht eine höhere Auslastung auf Kosten der Durchlaufzeit. Die Zusammensetzung der Produktionsaufträge in den drei Szenarien ist in Abbildung 8.2 gezeigt. Dort sind auch die entsprechenden Auslastungen und die Flussfaktoren aufgezeigt.

Wie in Kapitel 7.1 beschrieben, orientiert sich das Simulationmodell stark an einem realen Fertigungsabschnitt, entspricht diesem aber nicht exakt. Einige Gegebenheiten der realen Produktion ließen sich im Simulationsmodell nicht abbilden. Bei der simulierten Fertigungslinie wurde schlussendlich darauf geachtet, dass die Betriebspunkte und Variabilitäten in den Szenarien

mit der Realität übereinstimmen. Bei einer identischen Betriebskennlinie zweier komplexer Werkstattfertigungen wird davon ausgegangen, dass Ergebnisse und Auswirkungen des *Dispatchings* und *Schedulings* zwischen den Fertigungen übertragbar sind[1].

8.1.2. Benchmark-Heuristiken im Simulationsmodell

Jede Maschinengruppe wird von einer anderen, in der Halbleiterfertigung typischen Heuristik gesteuert. An der Maschinengruppe 1 werden die Lose zuerst nach Verfügbarkeit der Fotomaske gefiltert. Danach werden die verbleibenden Lose nach ODD angeordnet. Ist die Auftragsreihenfolge dadurch nicht eindeutig bestimmt, wird auf FIFO zurückgegriffen.

An der Maschinengruppe 2 wird eine Hierarchie von drei unterschiedlichen Heuristiken angewendet. Zuerst werden die Aufträge nach einer Rüstoptimierung eingeteilt, um die Rüstzeit zu minimieren. Danach werden die Aufträge nach ODD und danach nach FIFO angeordnet.

Da die Maschinengruppe 3 über eine unendliche Kapazität verfügt, werden alle Aufträge sofort gestartet. Daher sind keine *Dispatching*-Heuristiken notwendig.

An Maschinengruppe 4 wird der Batch mit der größten Abweichung vom Liefertermin gestartet. Dabei werden die Abweichungen der Aufträge addiert. Wird eine Zeitkopplung überschritten, wird das entsprechende Los sofort gestartet. Es wird dann ermittelt, ob es mit einem Auftrag zu einem Batch kombiniert werden kann. Ist dies nicht der Fall, wird der Auftrag alleine gestartet.

8.1.3. Wahl der Hyperparameter

Das Design der NNs ist einfach gehalten. Die NNs der DQN-Agenten im MAS sind alle gleich aufgebaut. Das NN besteht jeweils aus drei vollständig verbunden Ebenen (engl. *dense layers*). Die erste verdeckte Ebene hat 512 Neuronen, die zweite 128 und die dritte 18. Die Anzahl Neuronen der ersten

[1] Erfahrungswissen der Fachexperten der Infineon Technologies AG

zwei Ebenen wurden experimentell so gesetzt, dass das NN die Komplexität der Aufgabe abbilden kann. Gleichzeitig sollten die verdeckten Ebenen des NNs aufgrund der notwendigen Rechenzeit nicht zu groß gewählt werden. Die dritte Ebene entspricht der Ausgabe des *Q*-Wertes einer Aktion. Als Aktivierungsfunktionen werden in allen Ebenen ReLUs verwendet.

Die Wahl der Hyperparameter des DQN-Agent-Algorithmus ist in Tabellen 3.2 und 3.3 diskutiert. Die Wahl der Hyperparameter lässt sich auf zwei große Unterschiede – bzw. Herausforderungen – im Vergleich zu Mnih et al. (2015) im optimierenden Problem des RL zurückführen: Rechengeschwindigkeit und zeitlicher Einfluss einer Aktion. Aufgrund der geringeren verfügbaren Rechenleistung mussten einige Parameter so gewählt werden, dass eine Konvergenz in vertretbarer Zeit eintrat (siehe Spalte *Begründung für Veränderung* in Tabellen 3.2 und 3.3).

Der zeitliche Einfluss einer Aktion ist in der Umgebung in Mnih et al. (2015) entscheidend. In einem Atari-Spiel, in dem z.B. Rennen gefahren werden, ist sowohl die Richtung als auch die Dauer eines Lenkvorgangs entscheidend. Bei der *frame*-Rate und einer menschlichen Reaktionszeit ist davon auszugehen, dass die gleiche Aktion über mehrere *frames* ausgeführt wird. Die Situation von *frame* zu *frame* ändert sich kontinuierlich. Im Gegensatz dazu verändert jede Aktion in der Fabrikumgebung den Zustand so deutlich, dass das Wiederholen einer Aktion aufgrund des neuen Zustands nicht sinnvoll wäre. Daher wird in der Fabriksimulation zu jedem einzelnen Zustand eine explizite Aktion bestimmt.

8.1.4. Formulierung der Bewertungsfunktion

In diesem Experiment soll die Auslastung – und damit indirekt der Durchsatz – optimiert werden. Die Bewertung in Phase A und B wird wie in Kapitel 6.4 beschrieben unterschiedlich gegeben:

- **Bewertungsfunktion in Phase A:** Als Bewertung wird die Anzahl der Lose auf den Maschinen der Anlagengruppe geteilt durch die Kapazität verwendet. Die Kapazität bezieht sich auf die maximale Anzahl an Aufträgen, die simultan auf der Anlagengruppe bearbeitet werden

Hyper-parameter	Wert aus Literatur	Wert in dieser Arbeit	Begründung für Veränderung
agent history length	4	1	In der Erstveröffentlichung stellte eine Sequenz von mehreren Bildern einen Zustand dar. In dieser Arbeit ist ein Zustand eindeutig durch ein Abbild der Linie bestimmt.
target network update frequency	10 000 *(Hard-Update)*	10^{-2} *(Soft-Update)*	Es wurden Soft-Updates (<1) verwendet, da diese Sprünge im Lernverhalten verhindern. Ein Soft-Update von 10^{-2} ist äquivalent zu einem Hard-Update von 10^{2}. Der Wert des Soft-Updates wird möglichst groß gewählt, damit schneller gelernt wird. Wird er zu groß, wird der Lernvorgang instabil.
discount factor	0,99	0,9	Der *Discount*-Faktor γ bestimmt, inwiefern zukünftige Bewertungen mit einbezogen werden. Bei den Atari-Spielen in der Erstveröffentlichung musste auf das Ende des Spieles Bezug genommen werden, während in der Fabriksimulation eine kontinuierliche Berechnung der Bewertungsfunktion möglich ist. Der *Discount*-Faktor stellt hier sicher, dass die Fabrik kontinuierlich und langfristig stabil arbeitet.
action repeat	4	1	*Action repeat* spielt in der Fabrik keine Rolle, da jeder Schritt eine spezifische Entscheidung erfordert.

Tabelle 8.2.: Vergleich der Hyperparameter des DQN-Agenten-Algorithmus, Teil 1. Der Wert aus der Literatur bezieht sich auf Mnih et al. (2015).

Hyper-parameter	Wert aus Literatur	Wert in dieser Arbeit	Begründung für Veränderung
learning rate	0,00025	In dieser Arbeit wird der Adam Optimierer in Keras mit Standardeinstellungen anstatt des RMSProp Algorithmus verwendet. Phase A: *learning rate* = 10^{-4} Phase B: *learning rate* = 10^{-5}	
gradient momentum	0,95		
squared gradient momentum	0,95		
min squared gradient	0,01		
final exploration	0,1	0,3	Die finale Explorationsrate liegt höher, damit schneller gelernt werden kann.
final exploration frame	10^6	$5 \cdot 10^4$	Aufgrund der verfügbaren Rechenleistung, aber auch der hohen Informationsdichte im Zustand, wird die finale Explorationsrate früher eingestellt.
replay start size	50000	500	Der Parameter ist von hoher Bedeutung, wenn ein möglicher Endzustand in der Simulation bzw. dem Spiel schnell erreicht werden kann. Ein Endzustand existiert in der Fabrik nicht. Hier dient der Parameter dazu, den Zeitraum des Anlaufens der Linie auszuschließen.

Tabelle 8.3.: Vergleich der Hyperparameter des DQN-Agenten-Algorithmus, Teil 2. Der Wert aus der Literatur bezieht sich auf Mnih et al. (2015).

können. Dies entspricht der Auslastung der Maschinengruppe. Für nicht-valide Aktionen, die nicht ausgeführt werden können, wird von der Bewertung eine „Strafe“ von 1 subtrahiert. Ein Beispiel hierfür wäre die Verwendung eines Auftrags, dessen Fotomaske sich bereits in Benutzung befindet. Das Los wird dann nicht gestartet, da die benötigte Ressource nicht verfügbar ist. Eine weitere „Strafe“ wurde eingeführt, wenn 80 % des gesamten Umlaufbestands in der Linie an einer Maschinengruppe lagern. Tritt dieser Fall ein, wird eine Strafe von -2 gegeben und der nächste Schritt nicht über den DQN-Agenten, sondern über die Heuristiken bestimmt.

Die zweite „Strafe“ wird eingeführt, da der DQN-Agent beim Kennenlernen seiner Umgebung viele invalide Schritte ausführt. Um die Bestrafung durch invalide Schritte zu umgehen, ist eine mögliche Strategie, immer die Warteaktion auszuführen, bei der kein Auftrag gestartet wird (siehe Kapitel 6.2). Dieses Vorgehen wird durch die zweite „Strafe“ verhindert.

- **Bewertungsfunktion in Phase B:** Die Bewertungsfunktion wird nun auf die ganze Linie ausgedehnt. Es wird der Mittelwert der Auslastungen aller Maschinengruppen verwendet. Die Strafen werden beibehalten.

Im Anwendungsfall zur Messung der Linienleistung werden die „Strafen“ sowie die explorativen Aktionen gestrichen.

8.1.5. Resultate

Der Lernvorgang eines DQN-Agenten kann durch eine Reihe von Parametern überwacht werden, welche in Kapitel 3.2 erläutert wurden. Im Folgenden wird der Verlauf der Parameter während des Lernvorgangs diskutiert.

Zuerst werden die Parameter in Phase A betrachtet. Die Verlustfunktion eines NN gibt an, wie groß die Änderung des NN bei jedem Parameter-Update, also dem Lernvorgang, ist. Hier wird die mittlere quadratische Abweichung (engl. *Mean Squared Error (MSE)*) genutzt, die zwischen den

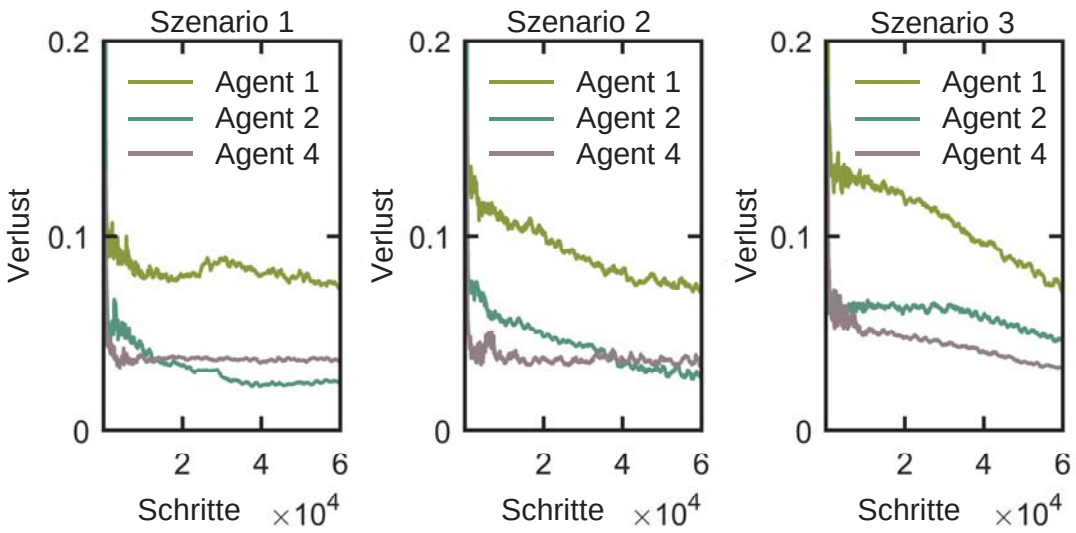


Abbildung 8.3.: Verlustfunktion der NNs in Phase A für die drei untersuchten Szenarien in Fallbeispiel 1 für alle drei Agenten

vom NN vorhergesagten und den tatsächlichen Werten ermittelt wird. Die Verlustfunktion der NNs für alle drei Szenarien ist in Abbildung 8.3 gezeigt. Alle Verlustfunktionen fallen, was auf eine Konvergenz gegen 0, bzw. einen kleinen Wert, hindeutet. Ein Wert von 0 würde bedeuten, dass das NN nicht mehr lernt, bzw. alle Trainingsdaten vollständig korrekt bearbeitet. Eine Konvergenz gegen einen kleinen Wert ungleich 0 ist zu erwarten, da in den Trainingsdaten durch die stochastischen Elemente in der Simulation immer ein Rauschen vorliegt. In Abbildung 8.3 konvergieren noch nicht alle Agenten eindeutig gegen einen Wert. Dies ist gewünscht, da der Lernvorgang in Phase B fortgesetzt werden soll.

Die Bewertungsfunktion in Phase A, hier die lokale Auslastung, wächst relativ schnell an, wie Abbildung 8.4 zu entnehmen ist. Dies liegt insbesondere daran, dass ohne Vorwissen gestartet wird. Das Verhalten der DQN-Agenten ist am Anfang zufällig, weshalb schnell deutliche Verbesserungen eintreten. Zudem hängt die erzielte Bewertung in der Simulation stark vom Anteil der explorativen Aktionen ab. Besonders in Szenario 1 wird dies deutlich: Die Verlustfunktion des NN konvergiert relativ schnell, während die Bewertungsfunktion weiter steigt. Dies liegt am linearen Verlauf der *ϵ-greedy policy*, die den Anteil der explorativen, also häufig nicht optimalen Aktionen

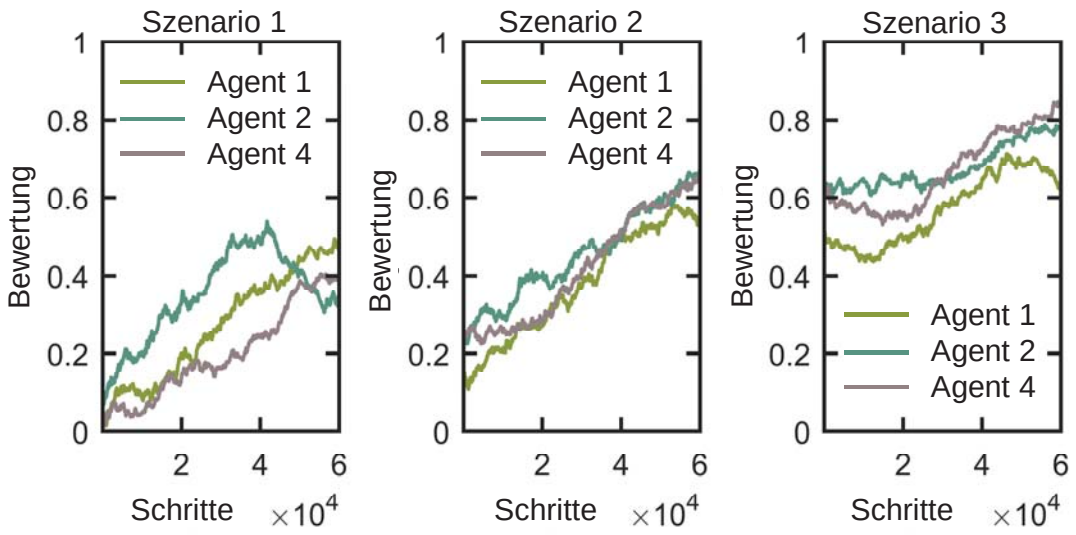


Abbildung 8.4.: Bewertungsfunktion in Phase A in den drei untersuchten Szenarien in Fallbeispiel 1

über die Zeit absenkt. In Szenario 1 tritt bei Agent 2 der Effekt auf, dass die Bewertungsfunktion gegen Ende wieder fällt. Dieser Effekt setzt sich in der folgenden Lernphase B allerdings nicht fort; dort steigt die Bewertung auch für Agent 2 wieder.

Während sich die Bewertung auf die tatsächlich erzielte Bewertung in der Simulation bezieht, wird bei der Q-Funktion in Abbildung 8.5 der vom NN vorhergesagt Wert aufgetragen. Die Q-Funktion ist die kumulierte, gewichtete Bewertung. Die Q-Funktion hat die Tendenz, am Anfang die Leistungsfähigkeit überzubewerten. Daher sinken die Kurven der Q-Funktion ab etwa $2 \cdot 10^4$ Schritten leicht. Je weniger explorative Aktionen eingestreut werden, desto besser kann das NN abschätzen, welche langfristige mittlere Bewertung es mit der entwickelten Strategie wirklich erreichen kann.

Generell überbewertet der DQN-Agent das Ergebnis seiner Steuerung. Bei einem Q-Wert von ~ 10 wird eine durchschnittliche Bewertung $\overline{r} = 1$ zugrunde gelegt:

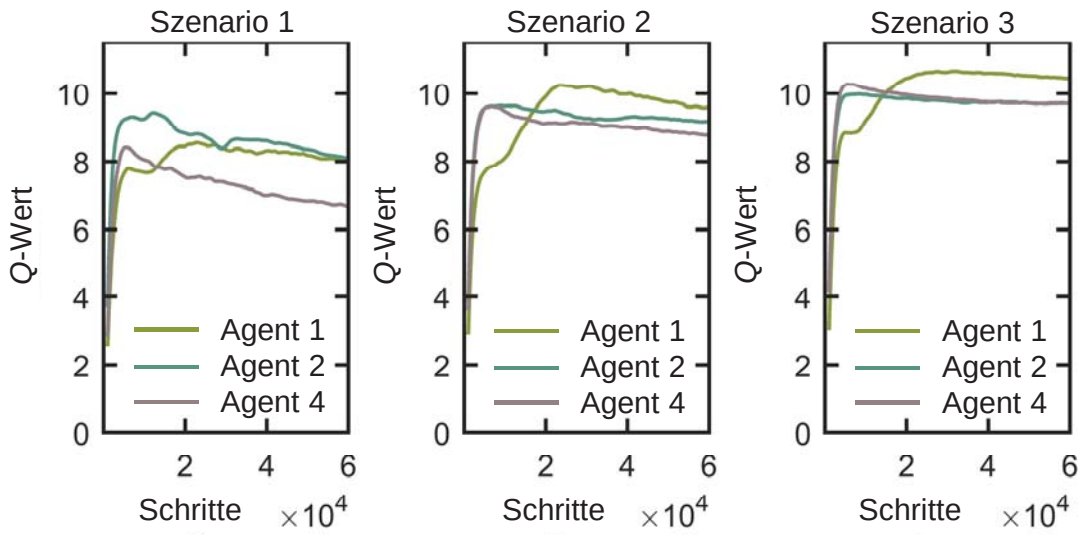


Abbildung 8.5.: Mittlerer Q-Wert während des Trainings in Phase A in den drei untersuchten Szenarien in Fallbeispiel 1

$$Q = 10 = \sum_{i=0}^{\infty} \overline{r} \cdot \gamma^i \tag{8.1}$$

$$\Rightarrow \overline{r} = \frac{10}{\sum_{i=0}^{\infty} \gamma^i} = 1, \tag{8.2}$$

wobei der *Discount*-Faktor γ der Q-Funktion im Experiment auf $0,9$ gesetzt ist. Damit überschätzt der DQN-Agent seine Leistungsfähigkeit, da eine Auslastung von 100 % nach der mathematischen Warteschlangentheorie in diesem System nicht erreicht werden kann (siehe Kapitel 2.4). Prinzipiell stellt dies in Phase A kein Problem dar. In Phase B wird jedoch eine korrekte Einschätzung und Konvergenz des Wertes erwartet.

In Abbildung 8.6 und Abbildung 8.7 sind Bewertung und Q-Wert in Phase B gezeigt. Es lässt sich feststellen, dass in Phase B die Leistungsfähigkeit deutlich verbessert werden kann. Die globale Koordination der DQN-Agenten hat also durchaus einen Einfluss. Die Q-Funktion bewegt sich auf ein realistisches Level, das mit den Ergebnissen in Tabelle 8.4 zusammenpasst.

Die Q-Funktion zeigt in Szenario 2 und 3 eine Konvergenz. In Szenario 1 ist dies nicht eindeutig der Fall. Hier könnte es sinnvoll sein, länger zu lernen.

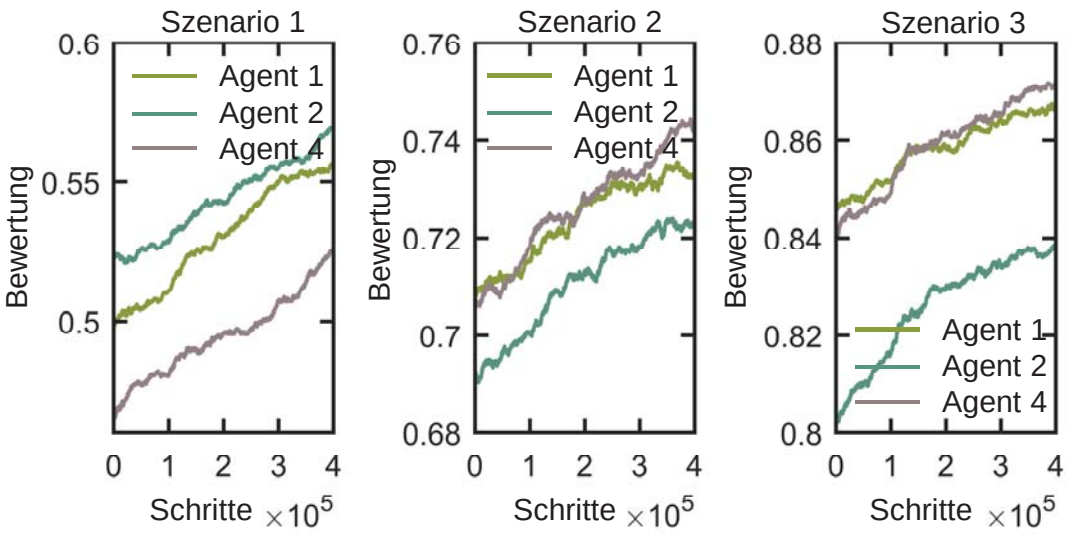


Abbildung 8.6.: Bewertungsfunktion in Phase B in den drei untersuchten Szenarien in Fallbeispiel 1

Allerdings wurde bei allen Szenarien der Lernvorgang zu einem festgelegten Zeitpunkt beendet, da nur eine begrenzte Rechenleistung zur Verfügung stand.

In Phase B wird allen DQN-Agenten die gleiche Bewertung bzw. der gleiche KPI gegeben. Trotzdem konvergieren die Kurven in Abbildung 8.7 nicht auf den gleichen Wert. Dies liegt daran, dass nur der lernende DQN-Agent explorative Aktionen einstreut. Der Einfluss von explorativen Aktionen auf die Linie ist bei allen DQN-Agenten unterschiedlich: Je nach Maschinengruppe führt eine zufällige Aktion zu unterschiedlichen Reaktionen. Daher konvergiert die Linienperformance bei den drei DQN-Agenten gegen unterschiedliche Werte.

Die endgültige Leistung der DQN-Agenten wird im Anwendungsmodus bewertet, in dem keine explorativen Aktionen und Änderungen der NNs vorkommen. Die Ergebnisse der Messungen sind in Tabelle 8.4 aufgeführt. Um eine bessere Einordnung der Ergebnisse zu ermöglichen, sind nicht nur die *Dispatching*-Heuristiken von Experten als Benchmark aufgeführt, die in Kapitel 8.1.1 vorgestellt wurden. Zusätzlich werden gestörte *Dispatching*-Heuristiken gezeigt, bei denen an Maschinengruppe 2 10 % bzw. 30 % der Aktionen zufällig ausgewählt werden. Der große Einfluss dieser Aktionen

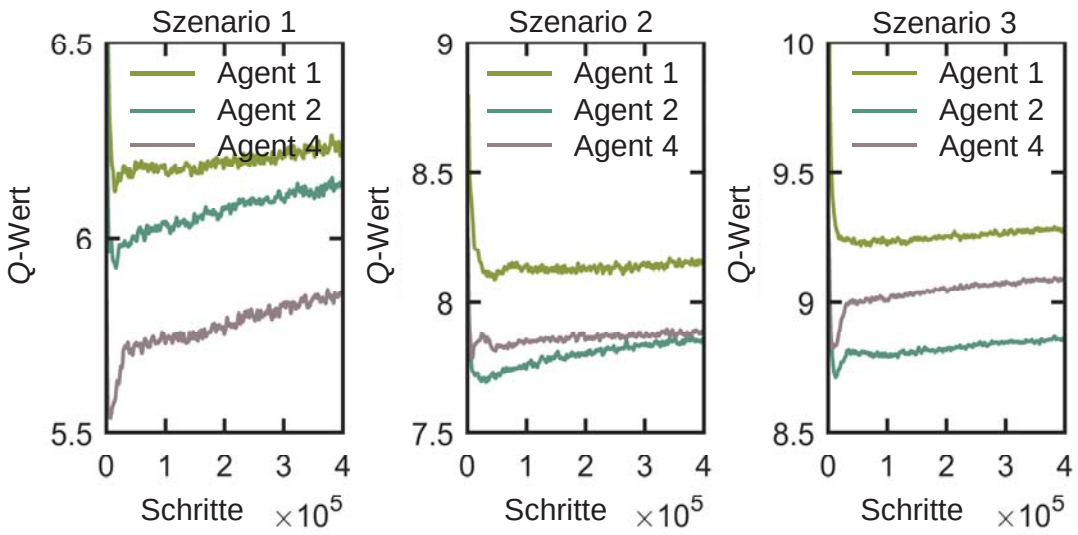


Abbildung 8.7.: Mittlerer *Q*-Wert während des Trainings in Phase B in den drei untersuchten Szenarien in Fallbeispiel 1

auf die Gesamtleistung der Linie zeigt, wie sensitiv das System auf Störungen reagiert. Gleichzeitig ist es ein Hinweis auf den Einfluss bzw. die hohe Bedeutung von *Dispatching*-Compliance, die in Kapitel 9 weiter diskutiert wird.

Die DQN-Agenten-Steuerung erreicht in Szenario 1 einen minimal besseren Wert als die Benchmark-Heuristik. In Szenario 2 liegen beide gleich auf und in Szenario 3 bleibt die DQN-Agenten-Steuerung leicht hinter der Heuristik zurück. Durch die 18 Neuronen in der Ausgabeebene des NN ist die maximale Warteschlangenlänge auf 17 Aufträge begrenzt. In Extremsituationen wird diese Länge in Szenario 3 durch den höheren Umlaufbestand überschritten. Die Warteschlange wird dann auf 17 Positionen beschränkt und das NN „sieht“ die überzähligen Lose nicht. Dadurch lässt sich die leichte Abweichung nach unten erklären.

Allgemein lässt sich jedoch sagen, dass das autonome *Dispatching*-System innerhalb einer Lernzeit von ca. 12 Stunden die Expertenregeln zur Produktionssteuerung selbständig finden konnte.

Für die simulierte Linie liegt kein Modell für eine mathematische Optimierung vor. Daher ist das absolute Optimum für die Liniensteuerung bzw. Linienleistung nicht bekannt. Allerdings erreichen sowohl die Experten-

Belohnung im Testmodus	Szenario 1	Szenario 2	Szenario 3
Benchmark-Heuristiken	0,61	0,83	0,96
Benchmark-Heuristiken mit einem explorativen Anteil von 10% an Maschinengruppe 2	0,58	0,79	0,91
Benchmark-Heuristiken mit einem explorativen Anteil von 30% an Maschinengruppe 2	0,55	0,73	0,85
Ergebnis der DQN-Agenten Optimierung	0,62	0,83	0,94

Tabelle 8.4.: Vergleich der DQN-Agenten-Steuerung mit den *Dispatching*-Heuristiken in der Testumgebung. Zur besseren Einschätzung werden zusätzlich Simulationsergebnisse gezeigt, bei denen 10 % bzw. 30 % der Aktionen an Maschinengruppe 2 zufällig gewählt werden. Die präsentierten Werte sind der Mittelwert über 25000 Schritte in der Simulation.

Heuristiken sowie die DQN-Agenten das gleiche Ergebnis. Dies legt nahe, dass es sich bezüglich der Auslastung um den optimalen Betriebspunkt handelt.

8.2. Fallbeispiel zur Liefertreue[1]

Der Aufbau des Kapitels zum zweiten Fallbeispiel folgt der Struktur des Kapitels des ersten Fallbeispiels.

8.2.1. Produktionslinie und Hyperparameter

Die Produktionslinie für das zweite Fallbeispiel ist ähnlich aufgebaut wie im ersten Fallbeispiel in Kapitel 8.1.1. Im Folgenden werden daher nur die Unterschiede zur Produktionslinie des ersten Fallbeispiels besprochen.

[1] Das Kapitel basiert auf der Veröffentlichung Waschneck et al. (2018b).

(willkürliche Zeiteinheiten)	TK1	TK2	TK3
TK1	0,00	0,11	0,19
TK2	0,41	0,00	0,32
TK3	0,13	0,30	0,00

Tabelle 8.5.: Rüstzeiten für das Umrüsten von einer Technologieklasse (TK) zur nächsten am *Implanter*

Die RPZs für die Prozesse wurden leicht verändert und sind in Abbildung 8.8 aufgeführt. Auch die Umrüstzeiten wurden, wie in Tabelle 8.5 gezeigt, angepasst.

Im Gegensatz zu Fallbeispiel 1 wird im zweiten Fallbeispiel nur mit einem Szenario gearbeitet. Die Unterschiede zwischen den drei Szenarien im ersten Fallbeispiel waren schlussendlich gering. Die Belegung der Produktionslinie des zweiten Fallbeispiels orientiert sich am Szenario 3. Von Technologieklasse 1 befinden sich 15 Lose im Umlaufbestand, von TK2 sind es 12 und von TK3 21 Aufträge.

Damit liegt in der Simulation ein realitätsnaher Flussfaktor von $FF = 2,8$ vor. Die Auslastung liegt bei 95 %, was dem Wert einer modernen Halbleiterfabrik entspricht.

Der prinzipielle Aufbau der Linie sowie die eingesetzten Heuristiken bleiben gleich wie in Fallbeispiel 1. Die komplette Produktionslinie für das zweite Fallbeispiel ist in Abbildung 8.8 gezeigt.

Auf die Wahl der Hyperparameter wurde im vorherigen Fallbeispiel in Kapitel 8.1.3 bereits eingehend eingegangen. Daher werden hier ebenfalls nur die Änderungen zu den Werten in Kapitel 8.1.3 diskutiert.

Beim NN wird die letzte Ebene auf 21 Neuronen erweitert, da der Umlaufbestand in Fallbeispiel 2 etwas höher liegt als in Szenario 3 im Fallbeispiel 2. Damit liegt die maximale Warteschlangenlänge bei 20 Aufträgen und nicht bei 17 wie in Fallbeispiel 1.

Der Parameter *learning rate* wurde von Phase B von 10^{-5} zu 10^{-6} geändert. Außerdem wurde der Parameter *minibatch size* in Phase B auf 8 herabgesetzt

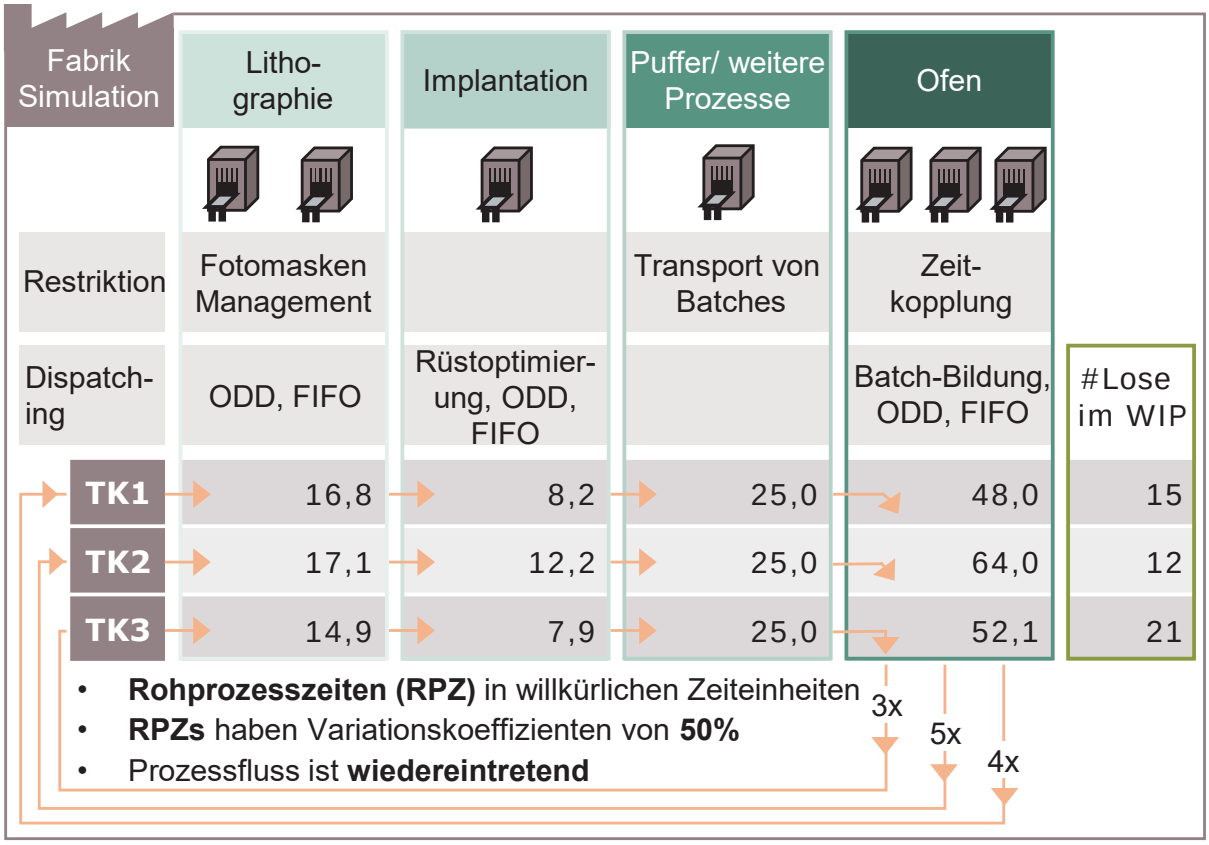


Abbildung 8.8.: Übersicht über das Simulationsmodell für Fallbeispiel 2 mit drei Technologieklassen (TKs) und vier Maschinengruppen. Die Anzahl der Aufträge in der Linie in der jeweiligen Technologieklasse sind links aufgeführt. Weitere Informationen sind in Kapitel 8.1.1 bzw. unter Abbildung 8.1 gegeben.

(siehe Tabelle 3.2). Beides wurde verändert, um in Phase B sensitiver auf globale Veränderungen zu reagieren. Der Parameter *final exploration* war bei 0.2 in diesem Fallbeispiel ausreichend.

Im Gegensatz zu Fallbeispiel 1 wurde im zweiten Fallbeispiel in Phasen A und B auch der *discount factor* γ unterschiedlich gesetzt. In Phase A ist der Wert auf $\gamma = 0,1$ gesetzt, in Phase B auf $\gamma = 0,9$. Diese Wahl bezweckt ein schnelleres Lernen in Phase A, da zeitlich nahe Effekte deutlich stärker gewichtet werden. Allerdings wird bei $\gamma = 0,1$ ein kurzfristiges Handeln belohnt, weshalb in der finalen Lernphase dann mit $\gamma = 0,9$ gearbeitet wird.

In Abbildung 8.9 ist der Verlauf der ϵ-*greedy policy* in Phase A gezeigt. Da der DQN-Agent ohne Vorwissen startet, liegt der Anteil der explorativen

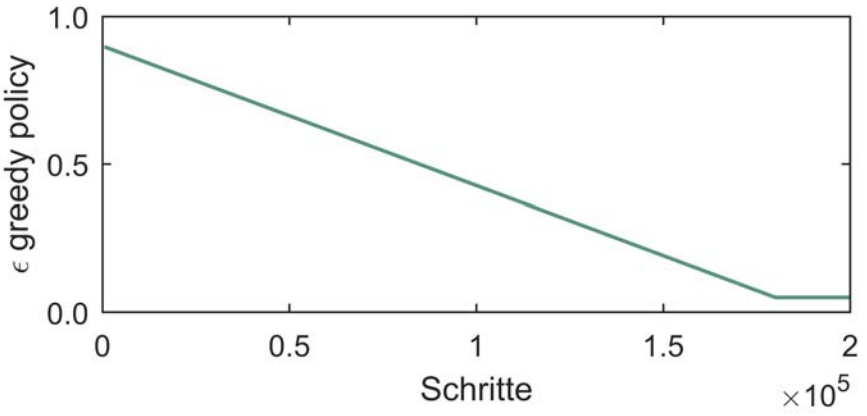


Abbildung 8.9.: Verlauf der *ϵ-greedy policy* in Phase A in Fallbeispiel 2

Aktionen anfangs bei 90 % und wird über den Verlauf des Lernprozesses kontinuierlich abgesenkt. Der Anteil der explorativen Aktionen beeinflusst auch den Verlauf der Bewertung, da bei einem hohen Anteil randomisierter Aktionen keine gute Leistungsfähigkeit erreicht werden kann.

8.2.2. Formulierung der Bewertungsfunktion

Die *Dispatching*-Heuristiken in der Simulation kombinieren die Ziele Auslastung und Liefertreue, wobei keine explizite Gewichtung vorgenommen wird. Die Auslastung wird insbesondere durch die Rüstoptimierung an der Implantation betont. Die Liefertreue wird durch die ODD-Heuristik gesteuert.

Wie bei ODD wird für die Bewertung der DQN-Agenten für jeden Produktionsschritt ein einzelnes *Due Date* errechnet. Der angewandte FF liegt bei $FF = 2,8$. Der FF erzwingt nicht nur eine Liefertreue. Ein solch hoher FF stellt sich erst bei hoher Auslastung ein.

Das Ziel dieses Fallbeispiels ist, die Zahl an Aufträgen mit einer Verspätung von ≥ 10 % zu reduzieren. Dies entspricht einem FF von $\geq 3,1$.

Die Bewertung in Phase A ist die normalisierte *Due-Date*-Abweichung des ausgewählten Auftrags. Das Netzwerk wird primär darauf trainiert, immer den Auftrag mit der lokal größten Abweichung zum Plantermin als nächsten zu verarbeiten. Dabei wird beispielsweise keine Rücksicht auf Rüstzeiten

genommen, sodass der Gesamtdurchsatz rückläufig sein könnte und zu einer generellen Verspätung aller Aufträge führen könnte.

Daher wird in Phase B eine globale Bewertung angewendet: Den DQN-Agenten wird die Summe der *Due-Date*-Abweichungen aller Lose in der Linie gegeben. Positive Abweichungen bei Aufträgen, die schneller als geplant durchlaufen, werden auf 0 gesetzt. Sie erzeugen zwar zusätzlichen Umlaufbestand, sind aber prinzipiell kein Problem für die Produktion. Trotzdem soll verhindert werden, dass verspätete Aufträge durch verfrühte ausgeglichen werden können.

Falls der DQN-Agent eine Aktion auswählt, die nicht ausgeführt werden kann, wird eine Strafe von -1 gegeben.

8.2.3. Resultate

Vor der Präsentation der Resultate werden die wichtigsten Parameter des Lernprozesses diskutiert.

In Abbildung 8.10 ist die Verlustfunktion der NNs gezeigt. Sie konvergiert für die Agenten 2 und 4 relativ schnell. Bei Agent 1 ist keine vollständige Konvergenz zu sehen. Da in Phase B weitergelernt wird, kann dies hingenommen werden. In Phase A soll schnell eine Prägung der NNs erreicht werden. Da in Phase B mit einer leicht veränderten Bewertungsfunktion trainiert wird, ist es nicht notwendig, dass die DQN-Agenten vollständig „angelernt“ sind.

In Abbildung 8.11 ist die Bewertungsfunktion gezeigt. Sie steigt konstant über den Lernvorgang an. Gegen Ende, bei einem konstanten ϵ, konvergieren die Bewertungsfunktionen der DQN-Agenten (siehe auch Abbildung 8.9). Damit wurde das Ziel von Phase A erreicht.

Zuletzt ist der Verlauf des vorhergesagten Q-Wertes in Abbildung 8.12 gezeigt. Auch dieser Wert konvergiert relativ schnell. Da der Q-Wert nicht die reale Bewertung, sondern die Vorhersage des NNs bei optimaler Steuerung repräsentiert, erkennt das NN das Potential relativ schnell. Die einfache Bewertungsfunktion in Phase A unterstützt die schnelle Konvergenz der Kurve.

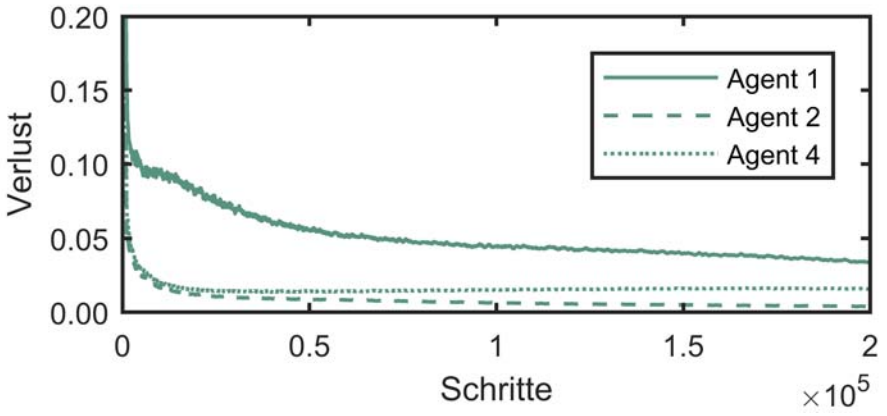


Abbildung 8.10.: Verlustfunktion der NNs in Phase A in Fallbeispiel 2

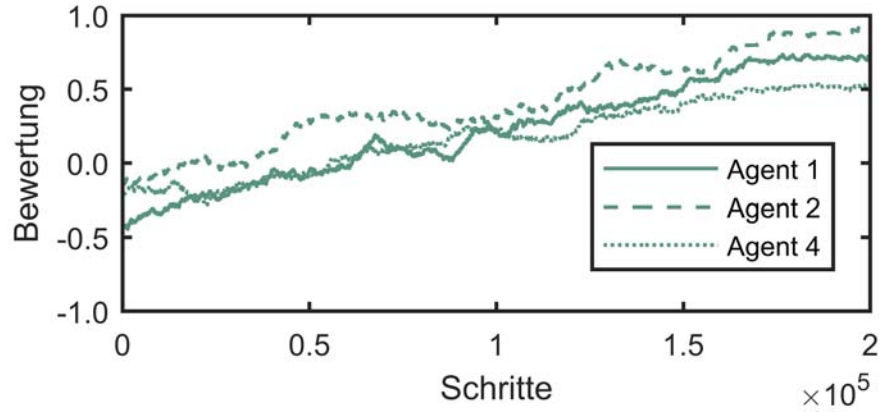


Abbildung 8.11.: Bewertungsfunktion in Phase A in Fallbeispiel 2

Der Verlauf der Kurven in Phase B ist nicht weiter aufgeführt. Sie verhalten sich ähnlich wie in Phase A bzw. in Fallbeispiel 1. Die Verlustfunktion konvergiert weiter, die Bewertung steigt im Mittel um weitere 5,1 % und der Q-Wert konvergiert ebenfalls.

Die finalen Ergebnisse wurden wieder im exploitativen Modus ermittelt, also ohne Lernen und explorative Aktionen. Die Resultate der Auswertung sind in Abbildung 8.13 und Tabelle 8.6 gezeigt. Die mittlere DZ für TK1 und TK3 ist nahezu identisch bei der konventionellen und autonomen Steuerung. Die mittlere DZ von TK2 wird sogar um 8 % angehoben. Die längere DZ eröffnet den DQN-Agenten nach dem *Operating Curve Management* die entsprechende Flexibilität um das eigentliche Ziel zu optimieren. Die Variabilität der

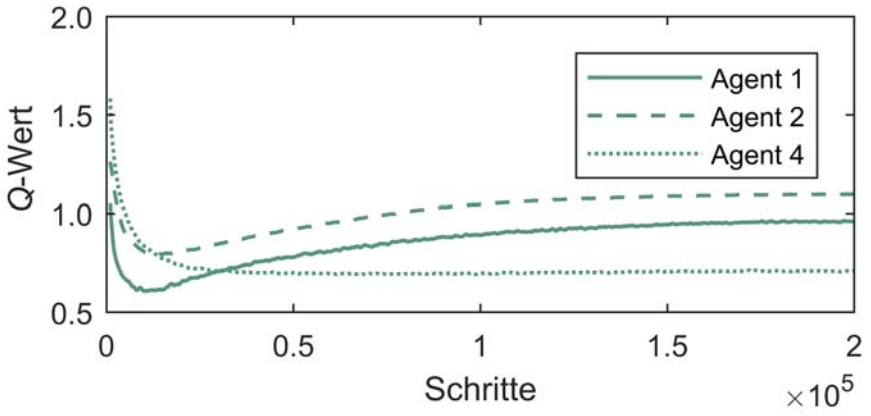


Abbildung 8.12.: Mittlerer Q-Wert während des Trainings in Phase A in Fallbeispiel 2

		TK1	TK2	TK3
Benchmark-Heuristiken	Mittlere Durchlaufzeit	1133,5	1600,1	838,0
	Standardabweichung der Durchlaufzeit	118,2	138,9	102,5
	Anteil Lose mit *FF* > 3,1	17,0%	1,7%	13,9%
DQN Agenten	Mittlere Durchlaufzeit	1118,2	1729,5	840,5
	Standardabweichung der Durchlaufzeit	56,2	60,9	54,5
	Anteil Lose mit *FF* > 3,1	1,3%	1,2%	1,9%

Tabelle 8.6.: Vergleich der wichtigsten Parameter der Liefertreue für eine Fabriksimulation in Bezug auf eine Steuerung durch *Dispatching*-Heuristiken und den DQN-Agenten-Ansatz. Die Werte beziehen sich auf 10^5 produzierte Aufträge.

Durchlaufzeit wird bei allen Technologieklassen im Mittel um 50 % gesenkt (siehe Tabelle 8.6). Der Anteil an Losen mit einer beträchtlichen Verspätung von über 10 % über dem Plan, also einem FF $\geq 3,1$ kann deutlich gesenkt werden. Am deutlichsten zeigt sich dieser Effekt in der TK1: Dort wird der Anteil der Lose mit FF $\geq 3,1$ von 17,0 % auf 1,3 % gesenkt werden. In TK3 fällt der Anteil immerhin von 13,9 % auf 1,9 %.

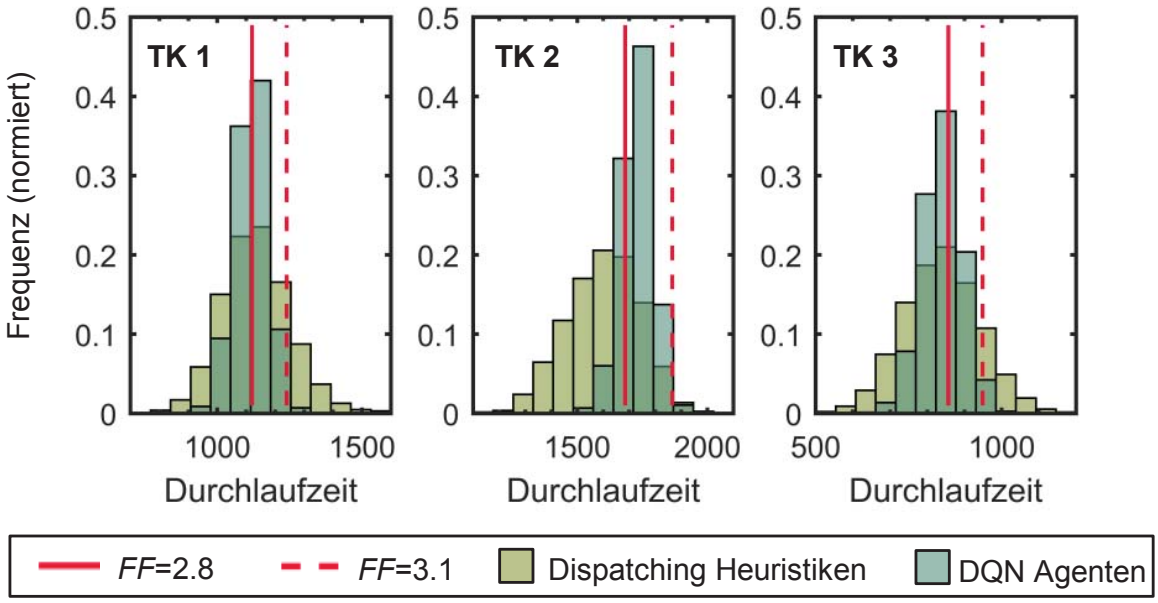


Abbildung 8.13.: Histogramme der DZ der produzierten Aufträge mit den entsprechenden *Dispatching*-Mechanismen. Die Werte beziehen sich jeweils auf 10^5 produzierte Aufträge, um eine entsprechende Statistik zu erreichen. Der Plan-FF beträgt $2,8$. Ab FF $= 3,1$ haben Aufträge eine kritische Verspätung.

Die autonome Produktionssteuerung konnte somit selbstständig eine Strategie finden, um die Zielfunktion zu optimieren. Diese übertrifft die Benchmark-Heuristik, die von Experten aufgesetzt wurde.

8.3. Untersuchung des Hyperparameterraums

Aufbauend auf Fallbeispiel 2 werden die entscheidenden Parameter des Lernprozesses verändert und die Auswirkungen beobachtet. Konkret wird die *learning rate (lr)* und die *target network update frequency (tnu)* jeweils um eine Größenordnung erhöht und gesenkt. Die anderen Hyperparameter werden dabei jeweils auf den Werten aus Fallbeispiel 2 fixiert.

Zuerst wird der Verlust des NNs betrachtet. Die gewählte Verlustfunktion, die mittlere quadratische Abweichung, ist der Standard im verwendeten ML Framework *keras*. Die Auswirkung der Hyperparameter ist in Abbildung 8.14 gezeigt. Bei dem Parameter *learning rate* ist deutlich, dass die Konvergenz des NNs während des Trainings stark von ihm abhängt. Bei einer zu kleinen *learning rate* konvergiert das NN zu langsam. Bei der vorhanden Rechenleistung muss daher eine größere *learning rate* akzeptiert werden, obwohl der Kurvenverlauf bei $lr = 10^{-5}$ glatter ist. Erstaunlicherweise beeinflusst die *target network update frequency* den Verlauf des Verlusts kaum.

Der Q-Wert wird ebenfalls von dem Parameter *learning rate* deutlich stärker beeinflusst als von der *target network update frequency*. In Abbildung 8.15 sind die Vergleiche gezeigt. Der Q-Wert konvergiert bei unterschiedlichen *learning rates* unterschiedlich schnell. Insbesondere bei Agent 1 zeigt sich, dass der Wert bei einer zu großen *learning rate* im Vergleich zu den kleineren *learning rates* zu früh auf einen schlechteren Wert konvergiert.

Zuletzt sind in Abbildung 8.16 die entsprechenden Bewertungen gezeigt. Dabei zeigt sich vor allem bei Agent 1, dass die gewählte Kombination der Hyperparameter zu den besten Ergebnissen führt. Allgemein sind die Unterschiede jedoch gering. Die Bewertungsfunktion wird in diesem Beispiel stärker durch den Verlauf der *linear annealed ϵ-greedy policy* beeinflusst, die den Anteil explorativer Aktionen steuert. Eine höhere Anzahl explorativer Aktionen brachte in diesem Fall keine weitere Verbesserung der Bewertung. Ein stärkerer Abfall in der *linear annealed ϵ-greedy policy* hätte den Lernvorgang beschleunigt, aber die Qualität der Ergebnisse verringert.

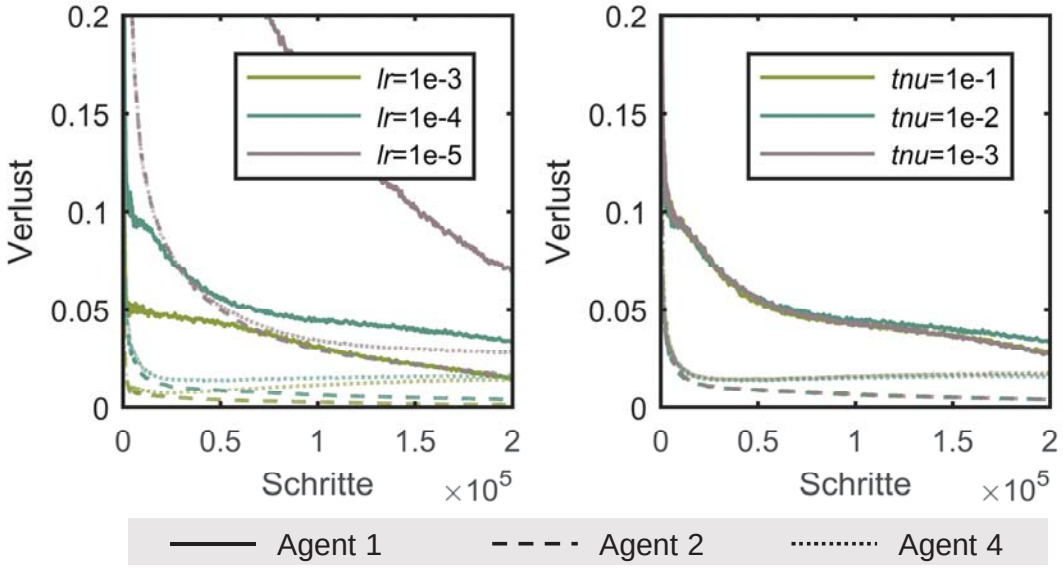


Abbildung 8.14.: Verlust des NNs während des Lernprozesses für unterschiedliche *learning-rate (lr)*- und *target-network-update-frequency (tnu)*-Parameter

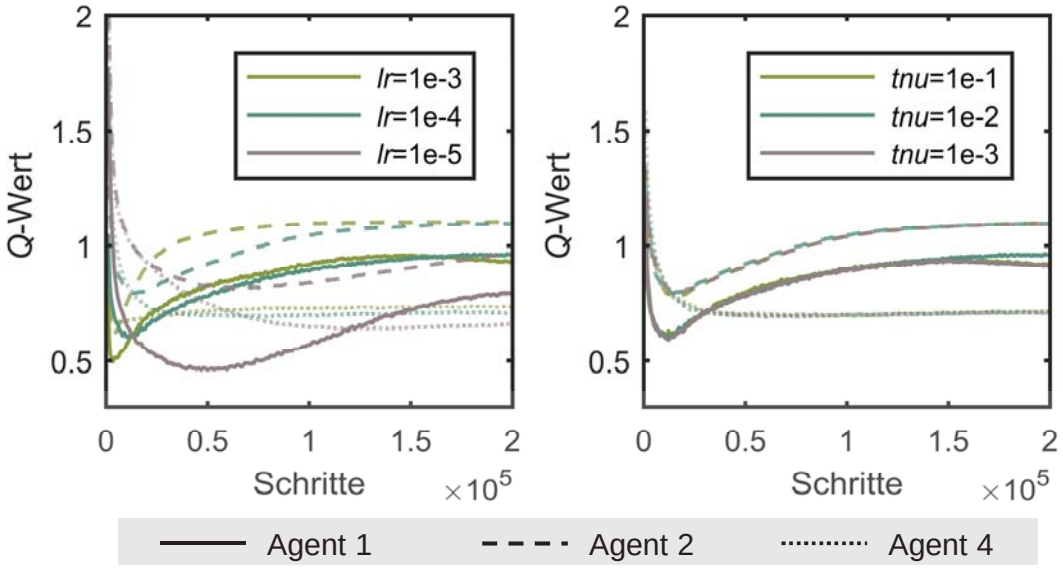


Abbildung 8.15.: *Q*-Wert der DQN-Agenten während des Lernprozesses für unterschiedliche *learning-rate (lr)*- und *target-network-update-frequency (tnu)*-Parameter

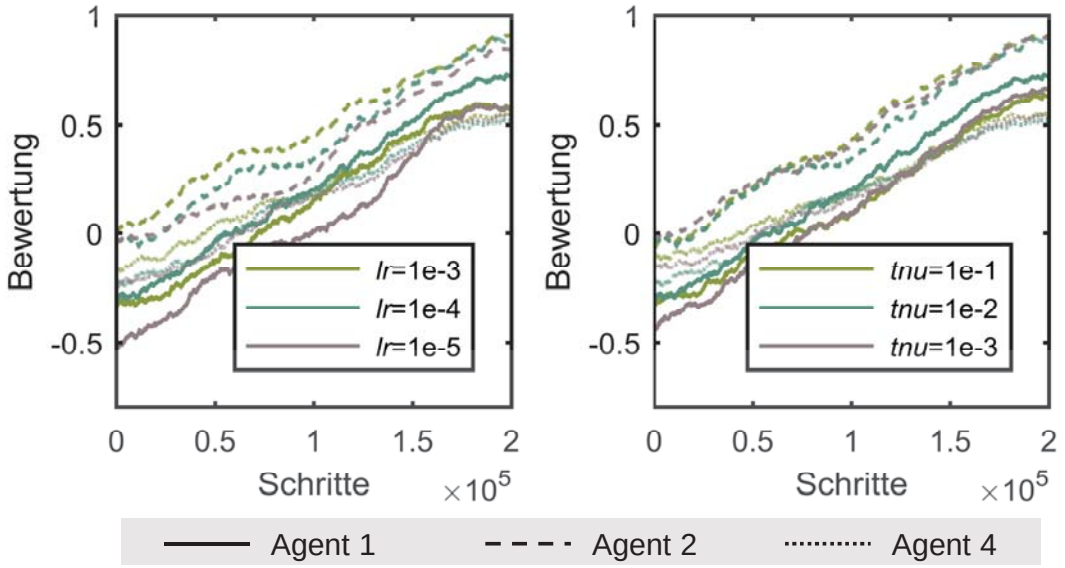


Abbildung 8.16.: Bewertung in der RL-Umgebung während des Lernprozesses für unterschiedliche *learning-rate (lr)*- und *target-network-update-frequency (tnu)*-Parameter

8.4. Kritische Bewertung

Nach der Anwendung der Methodik in zwei Fallbeispielen wird nun eine kritische Reflexion des autonomen *Dispatchings* vorgenommen. Darauf folgen allgemeinere Überlegungen zur KI innerhalb der Methodik.

8.4.1. Reflexion der Methodik

Die Methodik konnte in beiden Fallbeispielen angewandt werden und erzielte gute Ergebnisse bezogen auf die Zielgröße der Logistikleistung. Die in Kapitel 5.1 aufgestellten Bewertungskriterien reichen jedoch deutlich weiter. Im Folgenden soll die Methodik anhand der Anforderungen evaluiert werden.

- **Lokale Qualität der Ergebnisse:** Im Fallbeispiel zur Auslastungsoptimierung, Kapitel 8.1, konnte gezeigt werden, dass die Auslastung, die von der DQN-Agenten-Steuerung erreicht wird, ähnlich oder besser als lokale Heuristiken ist.
- **Echtzeitfähigkeit:** Die Methodik trennt den rechenaufwendigen Lernprozess vom Einsatz in der Produktion ab. In der Produktion wird das NN nicht mehr verändert und die Reihenfolgebewertung kann in Echtzeit über die Inferenz im NN bestimmt werden.
- **Globale Optimierung:** Im Fallbeispiel zur Liefertreue, Kapitel 8.2, konnte gezeigt werden, dass die DQN-Agenten in der Lage sind, lokale Kompromisse einzugehen, um die Gesamtleistung der Linie zu verbessern.
- **Transparenz in der Auswirkung auf Zielgrößen:** Im RL wird eine direkte, globale Bewertung formuliert. In der Simulation kann jederzeit bestimmt werden, inwiefern eine Zielerfüllung vorliegt.
- **Adaptivität:** Diese Methodik findet die Strategie zur Produktionssteuerung selbst. Diese kann automatisiert in die Produktion übertragen werden. Die Produktionssteuerung kann somit ohne Aufwand kontinuierlich angepasst werden.

- **Kontinuität:** Wie in Kapitel 6.3.1 beschrieben, erlaubt die Methode das Aufnehmen von bestehenden Produktionssteuerungen aufgrund der erfolgten Entscheidungen, also der MES-Produktionsdaten. Über kleine Lernraten und einen verkürzten Lernvorgang ist ein kontinuierlicher Übergang von einer bestehenden zu einer neuen Produktionssteuerung möglich.

Der Abgleich mit den in Kapitel 5.1 aufgestellten Anforderungen zeigt, dass die Methodik die identifizierte Forschungslücke erfolgreich schließt.

8.4.2. Reflexion bezüglich der Intelligenz des Systems

In Kapitel 3 folgt die vorliegende Arbeit Russell et al. (2010, S. viii) in der Definition, intelligente Agenten seien in der Lage, sich in unbekannten Umgebungen zurechtzufinden und zielgerichtet zu handeln. Nach dieser Definition kann festgestellt werden, dass sich die DQN-Agenten der autonomen Produktionssteuerung in der Fabriksimulation intelligent verhalten.

Allerdings meidet diese binäre Betrachtung der Intelligenz die aktuelle öffentliche Debatte über die Gefahren einer übermächtigen KI. Insbesondere Boström (2014, S. 194) argumentiert in seinem populären[1] Buch „Superintelligence: Paths, Dangers, Strategies“, dass KI die menschliche Intelligenz übertreffen und zur vorherrschenden Lebensform auf der Erde werden könnte. Größen des *Silicon Valley* wie Facebook-Gründer Zuckerberg und Tesla-CEO Musk befeuern die öffentliche Debatte, ob KI eine ernstzunehmende Gefahr (Musk) sei oder die Menschheit auf eine neue Stufe des Wohlstands heben werde (Zuckerberg; nach Metz, 2018).

Vor der breiteren Diskussion der Intelligenz des autonomen *Dispatchings* erfolgt ein kurzer Exkurs zu einer der populärsten Anwendungen der KI. *AlphaGo* (Silver et al., 2016) ist die erste Software, die einen professionellen *Go*-Spieler schlagen konnte. Die Weiterentwicklung *AlphaGo Zero* (Silver et al., 2017a) zeichnet sich dadurch aus, dass ohne menschliches Vorwissen ausschließlich durch RL trainiert wurde. Die Autoren um Silver berichten,

[1] Platz 17 der *Best Selling Science Books* der *The New York Times* am 8. September 2014

dass *AlphaGo Zero* Strategien entdeckte, die über das traditionelle *Go*-Wissen und bekannte Strategien hinausgehen.

Beim *Go*-Spiel haben Menschen schlussendlich das Nachsehen. Bei der Diskussion um Intelligenz sind allerdings andere Aspekte interessant. Erstens ist *AlphaGo* limitiert auf ein Brettspiel, und damit weit von einer generellen KI entfernt. Und zweitens lernten menschliche Spieler, die gegen *AlphaGo* spielten, von der Maschine und adaptierten ihre eigene Spielweise (Metz, 2016).

Gleiches gilt für das autonome *Dispatching*. Es ist in der Lage, neue Strategien in der Produktionssteuerung zu entwickeln. Diese sind in NNs kodiert und in dieser Form für den Menschen nicht verständlich. Aber durch Beobachtung der Entscheidungen des autonomen *Dispatching*-Systems lässt sich durchaus die Strategie der DQN-Agenten von Experten erkennen. Experten können also von dem System lernen. Jedoch kann der Mensch angesichts der Rechenleistung, die hinter dem autonomen *Dispatching* steht, nicht mithalten.

Turing formulierte 1950 die Idee, dass ein künstliches, intelligentes System für einen menschlichen Beobachter nicht unterscheidbar von einem Mensch sein dürfte (Turing, 2009). Heute ist diese Bewertung des Denkvermögens als Turing-Test bekannt. Die Prioritätsbewertungen des autonomen *Dispatchings* und die von Experten sind häufig identisch. Für einen Menschen ist nicht unterscheidbar, ob die *Dispatching*-Entscheidungen von einem Experten oder der KI des autonomen *Dispatchings* getroffen werden. Trotzdem bleibt die Einschränkung bezüglich der Intelligenz des Systems bestehen, da es im Gegensatz zum Menschen nur in einer spezifischen, klar definierten Aufgabe brilliert.

Hilfreich bei der Einordnung ist eine Unterscheidung zwischen starker und schwacher KI. Starke KI (engl. *strong AI* oder *artificial general intelligence (AGI)*) bezieht sich auf ein System, das über ein Bewusstsein und Empathie verfügt (Buxmann et al., 2019, S. 6; Pennachin et al., 2007, S. 1 ff.). Die starke KI zeichnet sich insbesondere dadurch aus, dass sie ihre Intelligenz auf jedes Problem anwenden kann. Ein solches System ist aktuell in der Forschung nicht bekannt (Buxmann et al., 2019, S. 6).

Eine schwache KI (engl. *weak AI* oder *narrow AI*) findet selbständig eine Lösung für eine abgegrenzte Aufgabe (engl. *narrow task*; Buxmann et al., 2019, S. 6 f.; Pennachin et al., 2007, S. 1 ff.). Sie scheitert allerdings an Problemstellungen, die vom ursprünglichen Problem abweichen. Die schwache KI ist also nicht in der Lage unterschiedliche Aufgaben zu lösen. Aktuell werden ausschließlich schwache KI Systeme praktisch eingesetzt (Buxmann et al., 2019, S. 6 f.; Pennachin et al., 2007, S. 1 ff.).

Nach dieser Unterscheidung ist das autonome *Dispatching* eindeutig ein typischer Vertreter der schwachen KI. Sie verfügt über kein Bewusstsein und keine Empathie, aber findet in einem fest definierten Aufgabenbereich, dem *Dispatching*, optimierte Strategien für vorgegebene Ziele.

Als schwache KI bedient das autonome *Dispatching* keine apokalyptischen Untergangszenarien, wie sie von Musk und Bostrom heraufbeschworen werden. Allerdings gefährdet es ganz konkret Arbeitsplätze: Die Stellen der Disponenten in der Produktion und der Regelschreiber, die die Heuristiken in der Produktion festlegen. Diese Diskussion ist real und sollte mit der entsprechenden Verantwortung und Ethik bei der Einführung eines intelligenten Systems geführt werden.

Kapitel 9

Soziotechnische Einordnung[1]

Seit der Pionierarbeit von Henry Gantt Anfang des 20. Jahrhunderts sind *Scheduling* und *Dispatching* als kritische Faktoren anerkannt, die maßgeblich die Leistung der Produktion bestimmen (Ouelhadj et al., 2009; Pinedo, 2016, S. vii). In den letzten etwa 70 Jahren wurden von Wissenschaftlern Lösungen für unzählige Optimierungsprobleme im Bereich *Scheduling* und *Dispatching* vorgeschlagen (siehe Pinedo, 2016). Trotzdem erleben Anwender regelmäßig signifikante Unterschiede zwischen der theoretischen Leistungsfähigkeit eines Algorithmus und den Ergebnissen der Anwendung in der Fertigung. Der Transfer von Algorithmen aus der Wissenschaft in die Fabrikhallen wird auch von der Wissenschaft als Herausforderung wahrgenommen (Mönch et al., 2011). Das autonome *Dispatching* bietet zwei Eigenschaften, die den Transfer vereinfachen sollen: das Anlernen des autonomen *Dispatchings* mit der aktuellen Produktionssteuerung (siehe Kapitel 6.3) und das Weiterlernen in der Produktion zum Ausgleich von Unterschieden

[1] Das Kapitel basiert auf der Veröffentlichung Waschneck et al. (2018a).

zwischen Simulation und Realität (siehe Kapitel 6.5). Trotzdem werden allgemeine Schwierigkeiten, die im folgenden Kapitel herausgearbeitet werden, auch beim Transfer des autonomen *Dispatchings* eine Rolle spielen.

Einerseits existieren in der Produktion oft äußerst produktspezifische Nebenbedingungen, denen beim Transfer nicht ausreichend Beachtung geschenkt wird, andererseits obliegt die Ausführung der Produktionsaufträge in der vorgegebenen Reihenfolge weiterhin meistens dem Werker. Bei der Algorithmen-Entwicklung werden menschliche Eigenschaften und Bedürfnisse häufig vernachlässigt. Pausen und Arbeitsgeschwindigkeit eines Menschen gehen meist nicht in die Algorithmenentwicklung ein. Abgesehen von Planungsvorschlägen, die durch einen Werker aufgrund von menschlichen Grenzen nicht ausgeführt werden können, kommt es auch zu technischen Limitationen. So können beispielsweise nicht alle Fehler der Maschinen automatisch vom System erfasst werden und in die Planung einfließen. Werker korrigieren diese Fehler in den Fertigungsbereichen. Trotzdem wird eine Abweichung vom Plan im System registriert. Schuld an diesen Abweichungen tragen meistens eine nicht ausreichende Datenerfassung und fehlerhafte Grunddaten.

In vollautomatisierten Halbleiterfabriken, meist mit einem Scheibendurchmesser von 300mm, spielen menschliche Faktoren eine untergeordnete Rolle. Allerdings gibt es weltweit 61 Unternehmen, die 200 mm Fabriken betreiben, jedoch nur 23 Firmen mit 300 mm Fertigungen[1] (McClean et al., 2018, Abbildung 4-20). Aufgrund des hohen nachträglichen Aufwands werden die meisten Scheibenfertigungen mit 200 mm nicht automatisiert[2], sondern werden weiter mit Werkern betrieben.

Im Folgenden sollen *Scheduling* und *Dispatching* in das soziotechnisches System eingeordnet werden. In einer Feldstudie beim Halbleiterhersteller Infineon Technologies AG wird die systemgenerierte Reihenfolge mit der tatsächlichen Ausführung verglichen. In den Fabriken des Unternehmens

[1] Allerdings machen 300 mm Fabriken 65,3 % der weltweit installierten Produktionskapazität aus, da sie für Volumenfertigung besser geeignet sind (McClean et al., 2018, Abbildung 4-19).

[2] Einschätzung der Infineon Technologies AG basierend auf Industriebenchmarks

werden sowohl die vorgeschlagene Reihenfolge als auch die tatsächliche Ausführung über *Radio-Frequency IDentification (RFID) Tags* digital erfasst und archiviert. Aus diesen Daten wird die *Dispatching*-Compliance errechnet.

Momentan werden für die Abweichung von der systemgenerierten Reihenfolge keine Begründungen aufgezeichnet. Daher ist es nicht direkt möglich, zwischen bewussten, begründeten Abweichungen der Werker und solchen Abweichungen zu unterscheiden, die sich negativ auf die Fabrikleistung auswirken. Da die Fabrikleistung von zu vielen Faktoren beeinflusst wird, war es nicht möglich „gute“ von „schlechter“ Non-Compliance zu unterscheiden.

9.1. Compliance in Halbleiterfabriken in der Literatur

In der Literatur finden sich wenige Diskussionen des Themas *Dispatching*-Compliance in Halbleiterfabriken. Die wenigen Referenzen geben einen Rahmen und eine Orientierung, um die folgenden Ergebnisse der Feldstudie einzuordnen.

Der Einfluss von *Dispatching*-Compliance kann direkt beobachtet werden. In der Simulation wurde dies in der vorliegenden Arbeit in Kapitel 8.1 gezeigt: Ein Anteil von nur 10 % zufälligen Produktionsaufträgen an einem Arbeitszentrum senkte die Auslastung der gesamten Produktionslinie um 5 %.

Chik et al. (2004) berichten von der Einführung eines *Dispatching*-Compliance-Überwachungssystems in einer malaysischen Halbleiterfabrik, die durch den Applied Materials Real-Time Dispatcher gesteuert wird. Nach der Einführung des Überwachungssystems konnte die DZ Variabilität in der Fabrik innerhalb eines halben Monats um 15 % verbessert werden (Chik et al., 2004).

Zum typischen Wert der *Dispatching*-Compliance in einer Halbleiterfabrik finden sich in der Literatur nur zwei Quellen. Girau et al. (2013) beschreiben die Einführung eines *Dispatching*-Systems in einer Halbleiterfabrik der X-FAB Silicon Foundries SE in Dresden. Die *Dispatching*-Compliance wird in

der Fabrik mit 55 % bis 68 % gemessen. Mehrere Gründe für die niedrige Compliance werden angeführt:

- Das manuelle Transportsystem ist nicht in das *Dispatching*-System integriert. Da der Losstandort dem Dispatcher nicht bekannt ist, sind Lose häufig nicht verfügbar.
- Werker bevorzugen Lose, die direkt an der Maschine gelagert werden, gegenüber Losen in entfernteren Lagern. Häufig wird *First-In-First-Out* angewendet.
- Die spezifischen Eigenschaften von *Cluster Tools* sind im *Dispatching*-System und MES nicht vollständig abgebildet. Häufig ist eine bestimmte Anordnung von Losen notwendig, die nur im Rezept des chemischen Prozesses vermerkt ist.
- Rüstzustände sind im *Dispatching*-System nicht für alle Tools erfasst.

Zur Verbesserung der *Dispatching*-Compliance werden Training für die Werker, eine veränderte Arbeitsorganisation sowie die Pflege und das Management der Datenquellen vorgeschlagen.

Chakravarthi wird in Chan (2016) mit der Aussage zitiert, dass sich die typische *Dispatching*-Compliance in manuellen Fabriken mit einem Waferdurchmesser von 200 mm zwischen 45 % und 85 % bewegt.

9.2. Relevante Fabrikinformationen und *Dispatching*-Regeln für die Fallstudie, Definition der *Dispatching*-Compliance

Bei der Halbleiterproduktion handelt es sich um eine komplexe Werkstattproduktion, die in Kapitel 2 beschrieben wird. Die Feinplanung wird aufgrund der langen DZ von > 30 Tagen und einer hohen Anzahl an Prozessschritten (> 500 EPAs) in Echtzeit durchgeführt.

Die Fabriken der Infineon Technologies AG weisen einen unterschiedlichen Automatisierungsgrad auf. In allen im Folgenden betrachteten Fabriken gibt es vollautomatisierte Arbeitsbereiche. Da Roboter immer nach den

Regeln arbeiten, sind diese Arbeitsbereiche von der Feldstudie nicht mit einbezogen. Von den fünf[1] *Frontend*-Standorten des Unternehmens werden zwei ausgeschlossen. In Dresden ist die Vollautomatisierung der Produktion umgesetzt, weshalb das manuelle Starten von Aufträgen entfällt. Der Standort Temecula im US-Bundesstaat Kalifornien wird aufgrund seiner vergleichsweise geringen Größe ausgeklammert. Aufgrund unterschiedlicher Skaleneffekte (engl. *economies of scale*) ist er nicht mit den anderen Standorten vergleichbar.

In der *Dispatching*-Liste werde die Lose zuerst nach ihren Prioritätsklassen angeordnet. Prioritätsklassen werden initial beim Auftragsstart, der Einschleusung in den Reinraum, vergeben. Sie können sich während des Produktionszeitraums ändern, wobei dies eher unüblich ist. Die wichtigsten Prioritätsklassen sind im Folgenden aufgeführt:

1. **ROCKET.** Höchste Prioritätsklasse für Eillose. Der Anteil an *ROCKET*-Losen in der Produktion liegt bei $< 0,1$ %. *ROCKET*-Lose werden nicht über das automatische Transportsystem, sondern direkt von Mitarbeitern transportiert.
2. **HOT.** Zweithöchste Prioritätsklasse für Eilaufträge. Der Anteil in der Produktion liegt bei < 1 %.
3. **TIMECOUPLING.** Gewisse Prozessabfolgen müssen auf unterschiedlichen Maschinen innerhalb eines gewissen Zeitfensters ausgeführt werden. Mit dem Start des ersten Prozesses in der Sequenz startet das Zeitfenster und Lose werden in die Prioritätsklasse *TIMECOUPLING* gehoben.
4. **DEV.** Entwicklung und operative Produktion laufen in der Halbleiterfertigung in derselben Produktionslinie. Entwicklungslose werden in einem beschleunigten Korridor produziert, um kurze Forschungs- und Entwicklungszyklen zu garantieren.
5. **PROD.** Stellt den Standardkorridor für Aufträge dar, die zum Kunden ausgeliefert werden.

[1] Stand April 2018, siehe *www.infineon.com*

6. **ANKO.** Die *Anlagenkontrolle (ANKO)* wird im besten Fall durchgeführt, wenn gerade keine operativen Aufträge an der Anlage warten. Wenn Anlagen definierte Zeiträume ohne Kontrolle überschreiten, überschreiben ANKOs die anderen Prioritätsklassen. Bei einer ANKO wird der Prozess auf ANKO-Wafern ausgeführt, und das Prozessergebnis danach in der Messtechnik kontrolliert.
7. **HOLD.** Die Produktion der Lose wird pausiert. Meistens ist die Entscheidung eines Ingenieurs notwendig, wie mit dem Auftrag weiter verfahren wird. Dieser entscheidet, ob bei einer beobachteten Abweichung an einem Kontrollpunkt weiter produziert, eine besondere Nacharbeit angeordnet oder das Los verworfen wird.

Innerhalb einer Prioritätsklasse werden die Lose nach Heuristiken angeordnet. Dabei wird meistens eine Hierarchie von Heuristiken angewandt. Ist die Anordnung der Lose nach einer Heuristik nicht eindeutig, wird für uneindeutige Sequenzen innerhalb der Warteschlange die nächste Heuristik angewandt. Für die Standardregel wird die Hierarchie der Heuristiken im Folgenden kurz erläutert:

1. **ODD.** Jedem Prozessschritt wird ein Fälligkeitsdatum gegeben. Die Abarbeitungsreihenfolge wird nach der absoluten Abweichung vom Plandatum für den jeweiligen Prozessschritt bestimmt. Das Los mit der größten negativen Abweichung, also Verspätung, wird bevorzugt. Die Fälligkeitstermine der Prozessschritte werden über einen festen Zeitraum pro EPA bestimmt, der sich aus der Multiplikation der RPZ mit dem FF ergibt.
2. **Critical Ratio (CR).** Bei CR wird die Zeitdifferenz zwischen Lieferdatum und aktuellem Datum durch die verbleibende Restbearbeitungszeit geteilt:

$$\mathrm{CR} = \frac{\text{Lieferdatum} - \text{aktuelles Datum}}{\text{Restbearbeitungszeit des Auftrags}} \tag{9.1}$$

Die Restbearbeitungszeit wird ebenfalls über RPZ und FF bestimmt. Aufträge mit einer CR unter 1 liegen hinter dem Zeitplan. Der Auftrag mit der geringsten CR hat höchste Priorität in der Abarbeitungsreihenfolge.

3. **First-In-First-Out.** Abarbeitungsreihenfolge in der Ankunftsreihenfolge.

An unterschiedlichen Maschinengruppen werden unterschiedliche Regeln und Hierarchien eingesetzt. Im Folgenden werden die wichtigsten Regeln beschrieben, die in der Feldstudie verglichen werden:

- **STAND.** Die STAND-Regel ordnet die Lose innerhalb einer Prioritätsklasse nach der oben beschriebenen Hierarchie der *Dispatching*-Regeln.
- **SESTD.** STAND-Regel, die zusätzlich die Rüstung beachtet und eine heuristik-basierte Rüstzeitminimierung durchführt.
- **BATCH.** Heuristiken oder mathematische Optimierung zur Batch-Bildung basierend auf gemeinsamen physikalischen Anforderungen an den Prozess. Die BATCH-Regel wird meistens an Öfen eingesetzt, an denen sechs Lose zu einem Batch zusammengefasst werden.
- **SETUP.** Mathematische Optimierung zur Reduktion von Rüstzeiten. Die Regel wird hauptsächlich an den Implantationsanlagen eingesetzt, deren Durchsatz maßgeblich von der Rüstzeitoptimierung abhängt.
- **NEXT3.** Bezieht sich auf die drei Prozessierungsschritte, die an einem Lithographie Cluster ausgeführt werden: Benetzung mit Fotolack, Belichtung und Entwicklung des Fotolacks. Die Fotomaske (engl. *Reticle*) muss an der Anlage verfügbar sein. Falls alle Nebenbedingungen erfüllt sind, wird ebenfalls die oben beschriebene Regelhierarchie verwendet.

Unter Compliance wird „regelgerechtes, vorschriftsgemäßes [und …] korrektes Verhalten“ verstanden (Dudenredaktion, 2018b). Unter *Dispatching*-Compliance wird die Einhaltung der systemgenerierten Reihenfolge der Produktionsaufträge durch die Werker verstanden.

Dispatching-Compliance sollte nicht als Bewertung der Arbeit des Werkers verstanden werden. Die *Dispatching*-Compliance bewertet die Güte des soziotechnischen Systems. Werkern wird explizit die Möglichkeit eingeräumt, von den Vorgaben des technischen Systems abzuweichen. Diese letzte Plausibilitätsprüfung ist beispielsweise sinnvoll, wenn bestimmte Maschinenfehler vom digitalen System / MES nicht erkannt werden. Mit dem vorliegenden Datensatz kann im Falle einer Abweichung nicht unterschieden werden, ob ein Systemfehler vorliegt oder die vom System vorgeschlagene Variante eine bessere Lösung wäre.

Die absolute Compliance C_{abs} bezeichnet die absolute Position des Loses in der *Dispatching*-Liste, wenn es für die Prozessierung auf der Maschine gestartet wird. Eine absolute Compliance von $C_{abs} = 3$ bedeutet, dass im System zwei Aufträge mit höherer Priorität vorhanden waren, als der Auftrag zur Prozessieren ausgewählt wurde. Es wird angestrebt, immer das Los an Position eins, also $C_{abs} = 1$, zu verwenden.

Die relative Compliance C_{rel} bezieht sich auf die relative Position des Loses in der Warteschlange:

$$C_{rel} = \begin{cases} 1 & \text{falls } l = 1 \\ 1 - \frac{(C_{abs}-1)}{(l-1)} & \text{falls } l \geq 2, \end{cases} \tag{9.2}$$

wobei l die Länge der Warteschlange bezeichnet. Eine relative Compliance $C_{rel} = 100\ \%$ bedeutet, dass das erste Los der Warteschlange gewählt wurde; bei $C_{rel} = 0\ \%$ wurde das letzte Los der *Dispatching*-Liste für die Prozessierung verwendet.

Außerdem wird teilweise eine binäre Compliance C_{bin} verwendet:

$$C_{bin}^{(n)} = \begin{cases} 1 & \text{falls } C_{abs} \leq n \\ 0 & \text{falls } C_{abs} > n, \end{cases} \tag{9.3}$$

wobei n die maximale Position angibt, die noch als regelkonform betrachtet wird. In dieser Feldstudie wird die binäre Compliance nicht verwendet. Da die Warteschlangenlänge nicht eingeht und der Parameter n frei gewählt werden

kann, ist die Interpretation des resultierenden Werts schwierig. Obwohl sie durch den binären Charakter (Ampel rot/grün) leicht in der Produktion eingesetzt werden kann, ist sie für eine differenzierte, wissenschaftliche Betrachtung weniger geeignet.

9.3. Ergebnisse der Feldstudie zur *Dispatching*-Compliance

Für die Feldstudie werden zwei Datensätze *D1* und *D2* verwendet. Es werden nur Aufträge aus dem *PROD* Korridor verwendet. Arbeitsbereiche mit Automatisierung werden ausgeschlossen.

Der erste Datensatz *D1* enthält die absolute Compliance und die Warteschlangenlänge aller Buchungen vom 1.–8. Oktober 2017. Dabei handelt es sich um $\sim 180\,\text{k}$ Buchungen am Standort Regensburg (Deutschland), $\sim 220\,\text{k}$ Buchungen am Standort Villach (Österreich) und $\sim 300\,\text{k}$ Buchungen am Standort Kulim (Malaysia). Die Anzahl der Buchungen skaliert mit der Standortgröße, der aktuellen Auslastung und dem Grad der Automatisierung.

Der zweite Datensatz *D2* besteht aus allen $\sim 4.400\,\text{k}$ Buchungen, die am Standort Villach im Jahr 2017 in manuellen Bereichen getätigt wurden.

Zuerst wird die Verteilung der Warteschlangenlängen analysiert. Abbildung 9.1 verrät bereits einiges über die Charakteristiken der Standorte. Am Standort Villach werden viele Kleinserien mit hoher technologischer Komplexität und Diversität produziert. Diese Charakteristik zeigt sich an der vergleichsweise großen Anzahl kurzer Warteschlangen mit ≤ 5 Losen. Daraus resultiert eine geringe Flexibilität, insbesondere im Vergleich mit den anderen Standorten.

Der Standort Kulim produziert weniger Technologien als die anderen Standorte, dafür aber mit einem größeren Volumen. Dadurch sind die Ressourcenpools für einzelne EPAs größer. Die längeren Warteschlangen erlauben eine größere Flexibilität und ein größeres Potential bei der Optimierung, beispielsweise bei der Vermeidung von Rüstzeiten. Der Standort Regensburg ordnet sich von der Charakteristik zwischen den Fabriken in Kulim und Villach ein.

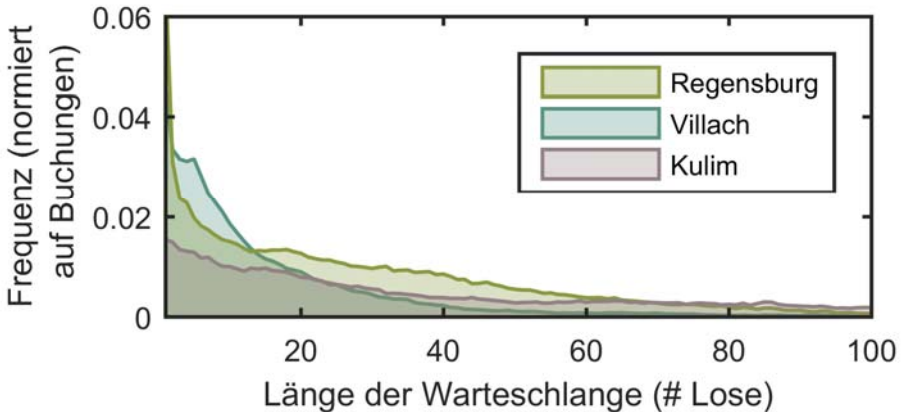


Abbildung 9.1.: Histogramm der Warteschlangenlänge der Standorte. Datensatz *D1*

Die absolute Compliance C_{abs} für eine bestimmte Warteschlangenlänge, die in Abbildung 9.2 gezeigt ist, ist an allen Standorten ähnlich. Für kurze Schlangenlängen ist die mittlere absolute Compliance natürlich geringer, d.h. besser, als für längere Warteschlangen. Dieser Effekt verschwindet, wenn die relative Compliance C_{rel} in Abbildung 9.3 betrachtet wird. Die relative Compliance beträgt in allen Fabriken im Mittel etwa 75 %. Die hohe relative Compliance für kurze Warteschlangen mit weniger als fünf Losen wird auch durch die Definition der Formel bzw. die stärker zutagetretende Quantisierung von C_{rel} bedingt. Bei nur einem Los in der Warteschlange beträgt die relative Compliance immer 100 %.

Am Standort Villach tritt bei Warteschlangen mit mehr als 90 Losen ein interessanter Effekt auf. Die relative Compliance fällt deutlich ab auf $\sim$ 50 %. Dies lässt sich dies mit der Auslegung des Standorts für kürzere Warteschlangen erklären (siehe Abbildung 9.1). Lagerplätze in der Umgebung der Maschinen sind nicht für die Menge an Umlaufbestand ausgelegt. Werker vermeiden dann längere Laufwege zu anderen Lagern oder das Suchen nach Aufträgen. Es werden Lose genommen, die schnell verfügbar sind, aber nicht unbedingt hohe Priorität haben. Meistens werden von Schichtleitern dann Ansagen gemacht, wie die KPIs *Wait-for-Load* und *Dispatching*-Compliance gegeneinander gewichtet werden sollen. Schlussendlich entscheidet jeder Werker bei jedem Los jedoch selbst, ob er ein schnell verfügbares Los wählt

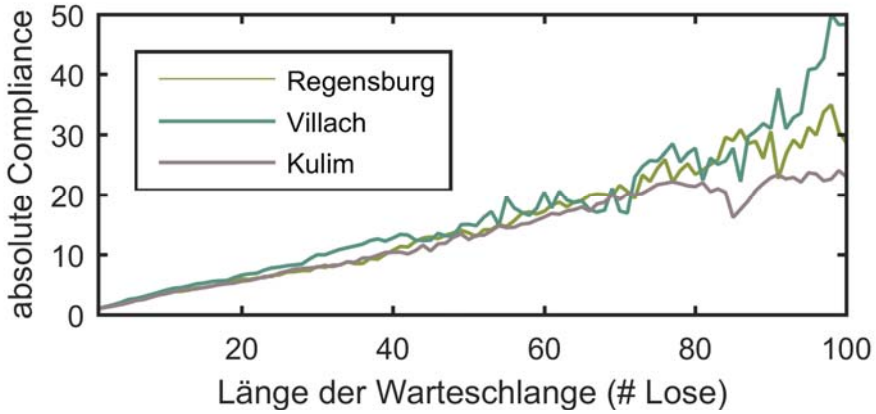


Abbildung 9.2.: Durchschnittliche absolute Compliance C_{abs} der unterschiedlichen Fabriken in Abhängigkeit von der Warteschlangenlänge. Datensatz *D1*

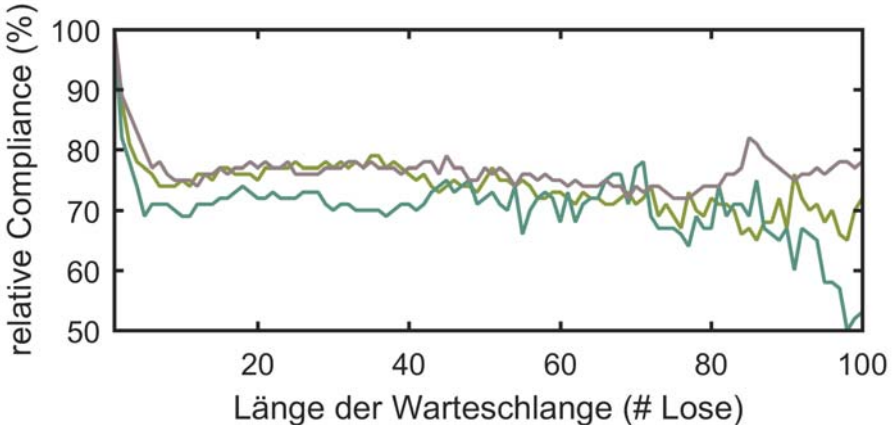


Abbildung 9.3.: Durchschnittliche relative Compliance der unterschiedlichen Fabriken in Abhängigkeit von der Warteschlangenlänge. Legende siehe Abbildung 9.2. Datensatz *D1*

oder eventuell das „richtige“ Los holt, dafür jedoch eine Stillstandszeit der *Bottleneck*-Maschine in Kauf nimmt.

Die zweite Begründung für die stärkere Abweichung ist eine Änderung der *Dispatching*-Strategie, die nicht im System implementiert ist. Lose durchlaufen jedes Arbeitszentrum mehrfach. Um den Umlaufbestand am Ressourcenpool schneller zu reduzieren, werden Lose bevorzugt, die zum letzten Mal dort einen Prozessierungsschritt durchlaufen. Lose, die kurz aufeinanderfolgend zwei oder mehr Prozessierungsschritte an diesem Arbeitszentrum

haben, werden herunterpriorisiert, da sie nur kurzfristig zu einer Entlastung führen.

Als Nächstes wird die relative Compliance der wichtigsten *Dispatching*-Regeln in der Fertigung in Abbildung 9.4 verglichen. In der Fabrik in Regensburg werden die STAND- und SESTD-Regel nicht unterschieden. Die einfache Rüstoptimierungsheuristik ist an allen Maschinen in der *Dispatching*-Regel enthalten, auch wenn keine Rüstung an der Maschine notwendig ist.

Die BATCH-Regel, die im Ofenbereich eingesetzt wird, zeigt an allen Standorten eine gute Compliance von deutlich über 90 %. Dies wird durch die Einfachheit der Regel und die relativ langen RPZs im Ofen-Bereich erklärt. Im Allgemeinen beträgt die Anzahl der vollständigen Batches von sechs Losen in einer Warteschlange nur ein bis zwei Batches.

Die ebenfalls gute Compliance der NEXT3-Regel wird mit der Kapitalintensität der Lithographie-Maschinen erklärt. Da eine Erweiterung der Kapazität mit einem vergleichsweise hohen Kapitalaufwand verbunden ist, liegt ein besonderer Fokus, insbesondere durch das Management, auf der guten Auslastung der Maschinen. Eine gute Auslastung wird durch die Kombination von durchdachten, angepassten *Dispatching*-Regeln und ihrer Einhaltung erreicht.

Der Durchsatz der Ionenimplantationsanlagen hängt maßgeblich von notwendigen Rüstvorgängen ab. Daher wird eine mathematische Optimierung verwendet, um die Rüstvorgänge und Rüstzeiten zu minimieren. Am Standort Regensburg wird eine Compliance von fast 100 % erreicht. Dies wird erklärt durch die kürzliche Einführung der SETUP-Regel und der damit verbundenen Aufmerksamkeit auf der Einhaltung der Regel durch Ingenieure.

Für den Vergleich der Compliance in unterschiedlichen Arbeitsbereichen wird Datensatz *D2* herangezogen. Die Warteschlangenlänge an den unterschiedlichen Arbeitsbereichen ist in Abbildung 9.5 gezeigt. Die Variabilität, gemessen zwischen der 25. und 75. Perzentile, ist in allen Arbeitsbereichen bis auf die letzten drei in etwa identisch. Maschinengruppen mit geringerem Kapitalbedarf in der Anschaffung haben im Normalfall etwas überschüssige Kapazität, um nicht zum Flaschenhals in der Produktion zu werden. Damit wird sichergestellt, dass kapitalintensive Anlagen ausgelastet sind.

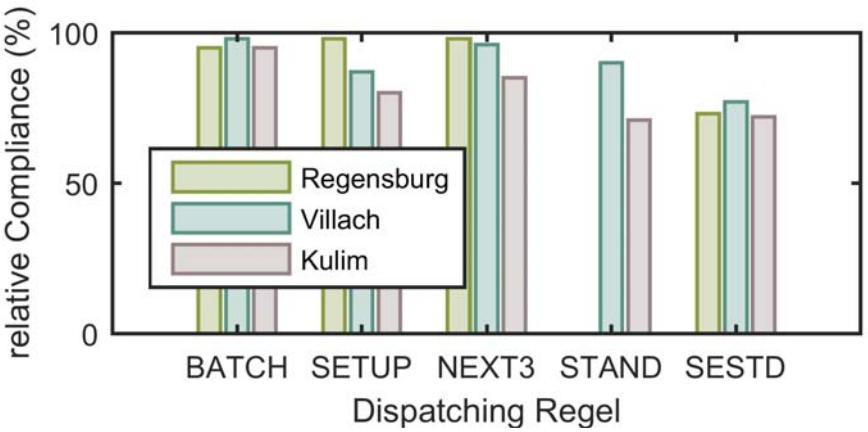


Abbildung 9.4.: Vergleich der mittleren relativen Compliance der Fabriken in Abhängigkeit der wichtigsten *Dispatching*-Regeln. Datensatz *D1*

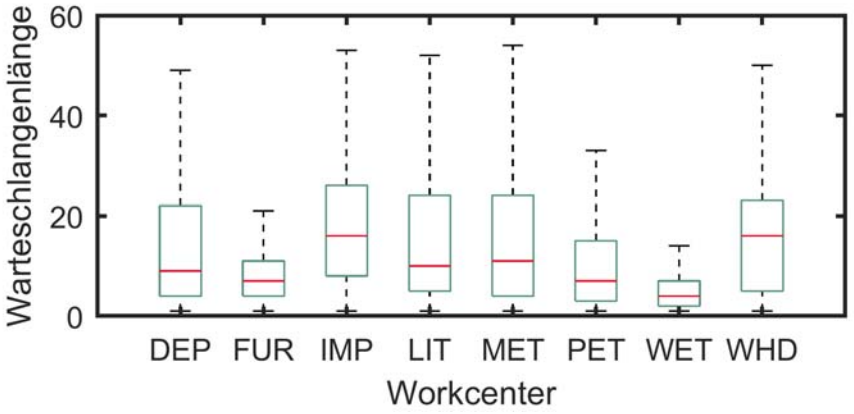


Abbildung 9.5.: Boxplot der Warteschlangenlänge an unterschiedlichen Maschinengruppen. Die rote Markierung ist der Median und die Ränder der Box sind das 25. Perzentil und 75. Perzentil. Die gestrichelten Ausläufer zeigen den Wertebereich der Daten an. Die Abkürzung der Maschinengruppen sind in Tabelle 9.1 erklärt. Datensatz *D2*

Die relative Compliance an unterschiedlichen Arbeitszentren ist in Abbildung 9.6 gezeigt. Wie in Abbildung 9.4 zeigen Maschinengruppen, die potentielle Flaschenhälse in der Produktion sind, eine gute relative Compliance. Dies sticht insbesondere bei der Lithographie und der Ionen-Implantation heraus. Die Verteilung der relativen Compliance ist an allen Arbeitszen-

Abkürzung	Englisch	Deutsch
DEP	Deposition	Abscheidung
FUR	Furnace	Ofentechnik
IMP	Implantation	Ionenimplantation
LIT	Lithography	Fotolithografie
MET	Metrology	Messtechnik
PET	Product Engineering Tools	Anlagen der Produktentwicklung
WET	Wet Chemistry	Nasschemie
WHD	Wafer Handling	Scheibenhandhabung

Tabelle 9.1.: Abkürzungsverzeichnis und Übersetzungen der Bezeichnungen der Maschinengruppen bei der Infineon Technologies AG. Die Abkürzungen sind typisch für die Halbleiterindustrie.

tren breit, was nahelegt, dass immer wieder Lose scheinbar „wahllos/zufällig“ ausgewählt werden.

Zuletzt werden die Autokorrelation und die Korrelationskoeffizienten bestimmt und in Abbildung 9.7 visualisiert. Ein absoluter Wert der Autokorrelation der relativen Compliance von $< 0,3$ zeigt, dass Effekte, die über mehrere Buchungen das Complianceverhalten beeinflussen, schwach sind. Der niedrige Korrelationskoeffizient der relativen Compliance mit der Warteschlangenlänge, die äquivalent zum Umlaufbestand im Bereich ist, zeigt, dass hoher Bestand nicht als allgemeine Erklärung für niedrige Compliance herangezogen werden kann. Auch mit den Zeitdifferenzen zwischen Buchungen, die reziprok mit dem Durchsatz skalieren, kann keine signifikante Korrelation festgestellt werden. Zusammenfassend können keine einzelnen, einfachen Begründungen für die *Dispatching*-Compliance herausgearbeitet werden.

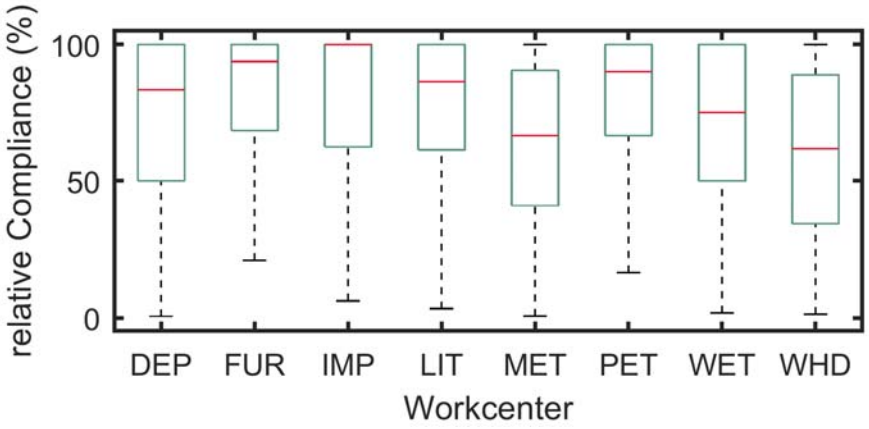


Abbildung 9.6.: Boxplot der relativen Compliance C_{rel} an unterschiedlichen Maschinengruppen. Die Abkürzung der Maschinengruppen sind in Tabelle 9.1 erklärt. Datensatz *D2*

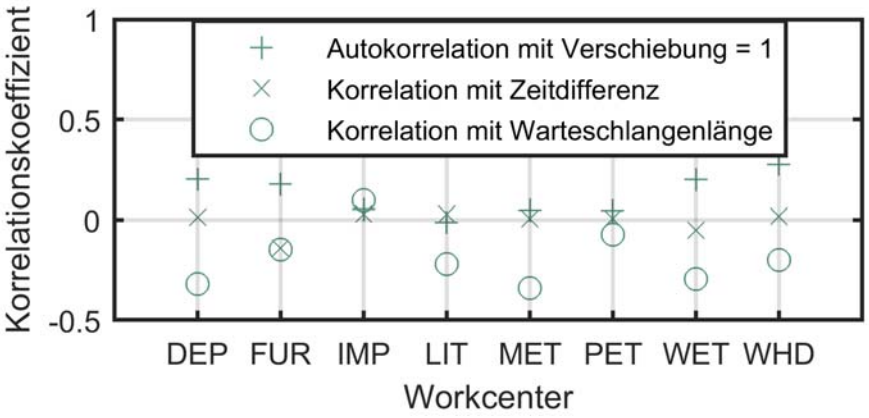


Abbildung 9.7.: Autokorrelation der *Dispatching*-Compliance mit Verschiebung um einen Zeitstempel (+) und Korrelationskoeffizienten an unterschiedlichen Maschinengruppen. Der erste Korrelationskoeffizient (×) bewertet die Korrelation zwischen relativer Compliance und den Zeitdifferenzen zwischen zwei Zeitstempeln, die reziprok mit dem Durchsatz skalieren. Der zweite Korrelationskoeffizient (∘) ist zwischen der relativen Compliance und der Warteschlangenlänge berechnet. Die Abkürzung der Maschinengruppen sind in Tabelle 9.1 erklärt. Datensatz *D2*

9.4. Diskussion der Ergebnisse der Feldstudie

Die Analyse hat gezeigt, dass unterschiedliche *Dispatching*-Regeln in unterschiedlichen Arbeitsbereichen die Compliance der Werker beeinflussen. Ein Haupteinflussfaktor, der auch in vielen Expertengesprächen mit dem industriellen Kooperationspartner der Feldstudie bestätigt wurde, ist die Aufmerksamkeit des Managements. Ist ein Engpass identifiziert, erhöht sich die Aufmerksamkeit des Managements, was zu einer höheren Compliance führt: Dies liegt sowohl an der ebenfalls erhöhten Aufmerksamkeit der Werker als auch an der Verbesserung und Behebung von Fehlern.

Obwohl keine klare Korrelation zwischen Umlaufbestand bzw. Durchsatz und Compliance in den Arbeitsbereichen gezeigt werden konnte, besteht trotzdem nach Expertenmeinung im Unternehmen der Feldstudie ein Zusammenhang. Vermutlich ist dieser komplexer und könnte in einer multivariaten Analyse mit weiteren Parametern aus anderen Datenquellen gezeigt werden. In Abbildung 9.3 konnte ein deutlicher Zusammenhang für lange Warteschlangen und Compliance am Standort Villach gezeigt und schlüssig erklärt werden.

Um die Compliance und damit die Fabrikleistung zu verbessern, sollte die Wichtigkeit von Compliance als KPI hervorgehoben werden. In flexiblen Werkstattfertigungen mit hohem Kapitalbedarf liegt der Fokus häufig auf der Minimierung von *Wait-for-Load*, also der Zeit zwischen der Bearbeitung aufeinanderfolgender Aufträge auf einer Maschine. Um die Auslastung zu maximieren, werden die Werker angewiesen, Aufträge schnell hintereinander zu starten. Um die Leerlaufzeit möglichst gering zu halten, werden dann Lose verwendet, die schnell auffindbar und nahe der Maschine sind. Trotzdem kann dieses Vorgehen den absoluten Durchsatz senken, da beispielsweise Rüstzeiten keine Beachtung finden.

Wenn *Dispatching*-Compliance verbessert werden soll, muss das Projekt von einem *Lean-Manufacturing*-Programm mit einem kontinuierlichen Verbesserungsprozess in der Produktion kombiniert werden. Wenn Werker von den computergenerierten *Dispatching*-Listen abweichen, ist es notwendig, eine Begründung dafür aufzuzeichnen, um die Regelqualität langfristig zu

verbessern. Selbst wenn die *Dispatching*-Regel nicht verbessert werden kann, beispielsweise aufgrund von Maschinenfehlern, die nicht digital erfasst werden können, kann die *Dispatching*-Compliance immerhin korrekt ermittelt werden, indem diese Fälle ausgeschlossen werden.

Menschliche Aspekte sollten also bereits in der Entwicklung von *Scheduling*- und *Dispatching*-Algorithmen stärker einfließen. Sowohl die Regel- und Datenqualität als auch das Verhalten der Werker spielen eine wichtige Rolle bei der Verbesserung der *Dispatching*-Compliance. Die nachhaltige Verbesserung der *Dispatching*-Compliance erfordert die Zusammenarbeit des Produktionsmanagements, der Ingenieure für Regelimplementierung, der Werker und der Forschung, die eine wichtige Rolle in der Algorithmenentwicklung spielt.

Bezüglich der Forschungsfrage muss damit zuerst eine Einschränkung gemacht werden: Eine optimale Produktionssteuerung garantiert noch keine verlustfreie Umsetzung eines optimalen Plans. Der Einfluss der Ausführung durch Werker darf nicht unterschätzt werden. Gleichzeitig stellen fehlerhafte Daten eine Schwierigkeit dar. Der Einführung des autonomen *Dispatchings* sollte daher erst ein kontinuierlicher Verbesserungsprozess mit Fokus auf Datenerfassung und Grunddaten vorausgehen.

Das autonome *Dispatching* basiert maßgeblich auf einer realitätsgetreuen Simulation. Diese wird aus den Stammdaten parametrisiert. Daher sollte vor der Einführung des autonomen *Dispatchings* die Grunddatenqualität geprüft werden. Die Simulation kann bereits vor der Einführung des autonomen *Dispatchings* mit der realen Produktion verglichen werden. Auftretende Abweichung sind ein Hinweis auf eine unzureichende Modellierung oder fehlerhafte Stammdaten.

Wie bereits am Anfang dieses Kapitels erwähnt, verfügt das autonome *Dispatching* über zwei Mechanismen, um die Einführung zu vereinfachen. Das autonome *Dispatching* kann, wie in Kapitel 6.3 beschrieben, mit bestehenden Produktionssteuerungen angelernt werden bzw. den Inhalt dieser aufnehmen. Daher gehen Verbesserungen der Steuerungsregeln, die vor der Einführung des autonomen *Dispatchings* implementiert wurden, nicht

verloren. Der Aufwand, die bestehende Steuerung zu überprüfen und zu verbessern, hilft direkt bei der Einführung des autonomen *Dispatchings*.

Der zweite Mechanismus greift nach Einführung des autonomen *Dispatchings*. Wie in Kapitel 6.5 beschrieben, kann die autonome Produktionssteuerung nach Einführung in der realen Fertigung weiter lernen. Damit können kleine Abweichungen zwischen Simulation und Realität korrigiert werden.

Ein genereller Vorteil des autonomen *Dispatchings* ist seine Flexibilität bei Veränderungen. Diese müssen nur in der Simulation implementiert werden. Danach optimiert das autonome *Dispatching* automatisch die Strategie der Abarbeitungsreihenfolge. Die damit angelernten NNs können dann direkt in der Produktion eingesetzt werden.

Kapitel 10

Zusammenfassung und Ausblick

10.1. Zusammenfassung

Komplexe Werkstattfertigungen benötigen eine Produktionssteuerung, die mit ihrer eigenen Dynamik und emergenten Phänomenen umgehen kann. Neue Methoden aus dem Bereich des ML bieten grundsätzlich das Potential und die Flexibilität, diesen Ansprüchen zu genügen. Im Forschungsprozess wurde erfolgreich eine selbstlernende, dezentrale Produktionssteuerung entwickelt.

Aufbauend auf den Grundlagen der Produktion sowie des ML wurden bestehende Ansätze zur Produktionssteuerung unter aktuellen Anforderungen bewertet. Dabei wurde deutlich, dass existierende Ansätze den gestiegenen Anforderungen der personalisierten Massenproduktion nicht gewachsen sind.

Der Forschungsprozess nach Ulrich et al. (2001, S. 195) weist einen hohen Praxisbezug auf. Diesem konnte durch zahlreiche Beispiele aus der

Halbleiterindustrie, insbesondere in Kapitel 4, Rechnung getragen werden. Durch den Kooperationspartner, die Infineon Technologies AG, wurde während der Untersuchung der Problematik immer die Anwendbarkeit und Praxisrelevanz sichergestellt.

Die Entwicklung einer Methodik zur autonomen Produktionssteuerung komplexer Werkstattfertigung beginnt mit dem RL-Ansatz. Es wurde ein Zustands- und Aktionsraum definiert und die Produktionssteuerung als Markow-Entscheidungsprozess formuliert. Eine effiziente Methode für das Training mit SL und RL wurde entwickelt. Durch die Verwendung von mehreren Agenten wird das schnelle Anlernen und die dezentrale Entscheidungsfindung unterstützt.

Ein Vorgehen für die Einführung in einer Halbleiterfabrik wurde in Kapitel 7 entwickelt. Mögliche Vorbehalte wie die Qualität des Simulationsmodells und die Interpretierbarkeit von NNs wurden besprochen.

Die Methodik konnte in zwei Fallbeispielen in Kapitel 8 validiert werden. Dabei wurden die gestellten Anforderungen an eine Produktionssteuerung einer komplexen Werkstattfertigung erfüllt. Im Fallbeispiel 1 konnte das autonome System innerhalb von zwei Tagen Rechenzeit eine Steuerungsregelung finden, die gleichauf mit den Expertenheuristiken lag. Im zweiten Fallbeispiel konnte der Anteil der verspäteten Lose durch eine verbesserte Steuerung von 17 % auf 1,3 % gesenkt werden. Dabei wurden der autonomen Steuerungsmethodik transparent Optimierungsziele vorgegeben, die innerhalb kürzester Zeit umgesetzt wurden.

Menschliche Aspekte bei der Produktion wurden in Kapitel 9 betrachtet. Während in der Arbeit eine langfristige technische Lösung diskutiert wurde, konnte in der soziotechnischen Einordnung auch ein kurzfristiger Vorschlag für die Verbesserung des Materialflusses gemacht werden. So lässt sich eine Erhöhung der *Dispatching*-Compliance bereits durch einen höheren Stellenwert des KPIs in der Produktionsstätte relativ einfach umsetzen.

Obwohl die Beispiele aus dem Bereich der Halbleiterindustrie stammen, ist die Methodik auf andere Industriezweige, die eine komplexe Werkstattfertigung nutzen, übertragbar.

In Bezug auf die Forschungsfrage ist festzustellen, dass die autonome Steuerungsmethodik den Umgang mit der zunehmenden Komplexität in der Werkstattfertigung signifikant unterstützt. Die Produktionssteuerung zeigt eine identische oder bessere Leistungsfähigkeit in Bezug auf die gesetzten Ziele als der Benchmark. Logistische Zielgrößen können flexibel und transparent gesetzt werden, denn die Steuerung passt sich automatisch an. Diese Eigenschaft ist insbesondere bei schwankender Nachfrage bzw. häufigen Veränderungen im Produktionsportfolio und Veränderungen in der Produktion, beispielsweise durch Kapazitätserweiterungen, bedeutend. Die Echtzeitfähigkeit garantiert außerdem in einer dynamischen Umgebung eine schnelle Entscheidungsfindung.

Bisher war die Regelerstellung und Optimierung eine manuelle Tätigkeit, die durch Experten erfolgte. Mit der autonomen Steuerung entfällt dieser langwierige manuelle Optimierungsprozess der Heuristiken bzw. Optimierungsmodelle. Die Produktionssteuerung wird autonom ohne menschliche Intervention an Veränderungen in der Produktion oder in den Logistikzielen angepasst.

10.2. Ausblick

Die Entwicklung und der Einsatz von KI-Methoden in der Produktionssteuerung stehen noch am Anfang. Das Forschungsfeld bietet in den nächsten Jahren ein enormes Potential, was insbesondere durch die neuen Möglichkeiten und rasanten Entwicklungen im Bereich KI begünstigt wird (LeCun et al., 2015).

Aktuell limitiert die verfügbare Rechenleistung den Einsatz von ML- und DL-Methoden in vielen Bereichen. Auch die Methodik dieser Arbeit war in den Fallbeispielen dadurch begrenzt. Die Entwicklungen in diesem Bereich sind jedoch vielversprechend. Das Moore'sche Gesetz besagt, dass sich die Rechenleistung eines Prozessors alle 18 Monate verdoppelt (Brock et al., 2006, S. 25 ff., Moore 1965). Im KI-Bereich wird dieses Tempo aktuell jedoch deutlich übertroffen: Amodei et al. (2018) ermittelten aus Veröffent-

lichungen zwischen 2012 und 2018, dass die verwendete Rechenleistung in den Publikationen sich alle 3,5 Monate verdoppelte. Seit 2012 beobachteten sie einen Anstieg in der Rechenleistung um einen Faktor von 300000. Dieses rasante Wachstum wurde durch dedizierte Hardware für DL und Software-Parallelisierung erreicht. Für die autonome Produktionssteuerung wird dieses enorme Wachstum neue Möglichkeiten eröffnen.

Die vorgestellte Methodik kann in mehrere Richtungen weiterentwickelt werden. Wie in Kapitel 3.2.3 geschrieben, existieren bereits diverse Modifizierungen des DQN-Agenten, die in dieser Methodik übernommen werden könnten. Die neueren RL-Agenten, wie der *Rainbow*-Agent (Hessel et al., 2018), sind in der Lage, schneller komplexere Zusammenhänge zu erlernen. Des Weiteren sieht der Autor Potential bei dem MAS-Ansatz, bei dem eine Kommunikation zwischen den Agenten expliziter ausgestaltet werden könnte.

Interessant wird die Anwendung auf verschiedene Skalierungen und Industrien. Bei der Skalierung und Parallelisierung des Trainingsprozesses ergeben sich spannende Fragestellungen für die Informatik. Durch veränderte Randbedingungen in unterschiedlichen Industrien können neue Ausgestaltungen des Zustandsraumes notwendig werden, die in dieser Arbeit nicht erfasst sind.

Die Voraussetzung für die Einführung des autonomen *Dispatchings* ist ein realitätsnahes Simulationsmodell oder ein Digitaler Zwilling. In einer Studie der Universität St. Gallen gaben nur 24 % der befragten Unternehmen[1] an, über eine Implementierung eines Digitalen Zwillings zu verfügen oder daran zu arbeiten (Friedli et al., 2018). In der Industrie stellt die Voraussetzung eines Digitalen Zwillings damit ein mögliches Hindernis für die Einführung des autonomen *Dispatchings* dar. Allerdings geben in der Studie 39 % der befragten Unternehmen an, die Technologie zu beobachten oder daran zu forschen. Daher ist zu erwarten, dass das Hindernis der Verfügbarkeit eines Digitalen Zwillings langfristig an Bedeutung verliert.

[1] Es beteiligten sich 137 Firmen in Deutschland und der Schweiz. Über ein Fünftel der Unternehmen haben mehr als 2000 Mitarbeiter, und etwa 20 % der teilnehmenden Firmen haben unter 50 Mitarbeiter (Friedli et al., 2018).

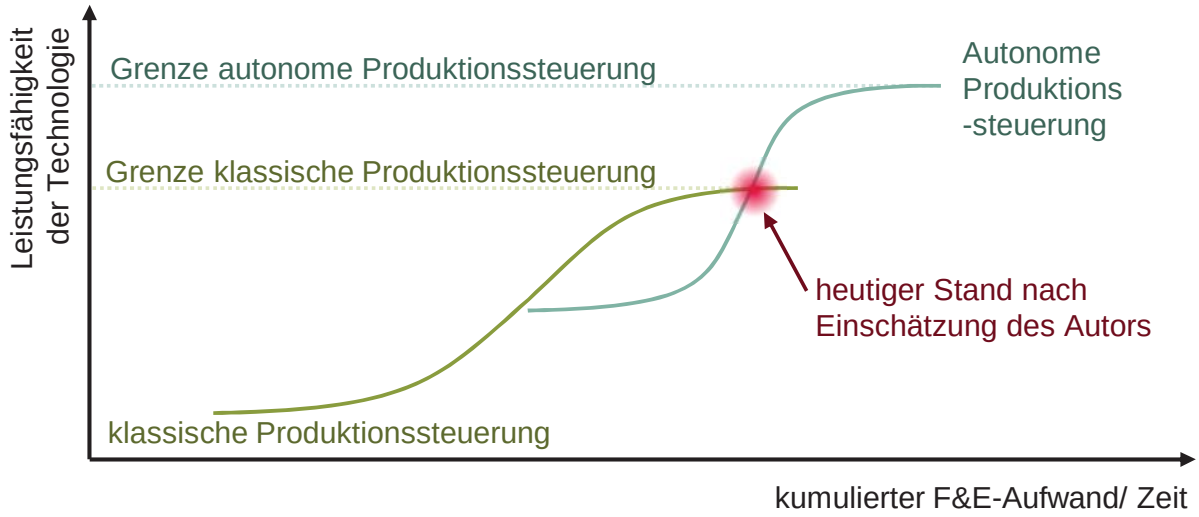


Abbildung 10.1.: Einordnung der autonomen Produktionssteuerung im S-Kurven-Konzept. Basierend auf Möhrle et al. (2018b)

Bei der Leistungsfähigkeit beginnt eine Substitutionstechnologie im Normalfall deutlich unter der etablierten Lösung. Die entsprechende Theorie ist als *S-Kurven-Konzept* bekannt (Möhrle et al., 2018b) und in Abbildung 10.1 gezeigt. Nach Einschätzung des Autors befindet sich die Entwicklung der autonomen Produktionssteuerung bereits am Schnittpunkt mit der bestehenden Lösung. Einerseits ist die hohe Leistungsfähigkeit des neuen Ansatzes demonstriert, andererseits herrscht aktuell eine Knappheit an KI-Experten, die entsprechende Konzepte implementieren könnten (Kahn, 2018). Bei der Entwicklung standardisierter ML-*Frameworks* und der Bereitstellung von Rechenleistung entwickelt sich das Angebot, wie oben beschrieben, enorm schnell. Der Autor dieser Arbeit ist überzeugt, dass sich ein autonomes Konzept der Produktionssteuerung mit ML-Elementen langfristig durchsetzen wird.

Anhang A

Informationen zum Autor

A.1. Tabellarischer Lebenslauf

Persönliche Daten

Name	Bernd Waschneck
Geburt	11.03.1988 in Heidelberg
Familien–stand	verheiratet
Wohnort	Neubiberg
Kontakt	bernd@waschneck.com

Ausbildung

2014 – 2018	Doktorand an der Graduate School advanced Manufacturing Engineering (GSaME), Universität Stuttgart. Erstbetreuer: Prof. Dr.-Ing. T. Bauernhansl
2011 – 2014	*Honours Degree in Technology Management* im bayrischer Elitestudiengang *„Technology Management“* am *Center for Digital Technology and Management (CDTM)*, einer Kooperation von LMU und TUM
2011 – 2014	Master of Science im Fach Physik an der LMU München, Schwerpunkt Nanophysik; Titel der Abschlussarbeit: *„Experimental Investigation of Quantum Point Contacts with Transport and Optical Spectroscopy“*
2011	Auslandssemester an der *National University of Singapore (NUS)*
2008 – 2011	Bachelor of Science im Fach Physik an der LMU München; Titel der Abschlussarbeit: *„Analysis of FDTD Discretization of Maxwell's Equations on non-uniform Grids“*

Berufserfahrung

2018	Infineon Technologies AG, Algorithm and Modelling Expert
2014– 2018	Infineon Technologies AG, Doktorand
2013	Startup DANATO GmbH, Studentische Unternehmensberatung am CDTM
2013	Praktikant bei Infosys Ltd. in Bangalore, Indien
2012 – 2013	Startup Preisenergie GmbH; *Managing Product Development* Kurs am CDTM
2010 – 2012	LMU München, Tutor der Vorlesungen „Rechenmethoden“ (Wintersemester 10/11) und „Theoretische Mechanik“ (Sommersemester 12)

A.2. Publikationen des Autors

Schubert et al., 2014

Schubert, Enrico; Heyder, Jan; Bauer, Florian; Waschneck, Bernd; Stumpf, Wolfgang; Wegscheider, Werner; Delft, Jan von; Ludwig, Stefan und Högele, Alexander, 2014.
Toward combined transport and optical studies of the 0.7-anomaly in a quantum point contact.
physica status solidi (b), **251** (9), S. 1931–1937.
DOI: 10.1002/pssb.201350218

Waschneck et al., 2016a

Waschneck, Bernd; Altenmüller, Thomas; Bauernhansl, Thomas und Kyek, Andreas, 2016a.
Production Scheduling in Complex Job Shops from an Industry 4.0 Perspective: A Review and Challenges in the Semiconductor Industry.
In: *CEUR Workshop Proceedings: Science, Application and Methods in Industry 4.0 (SAMI) @i-KNOW*.
Graz, Oesterreich, 19. 10. 2016, S. 3.1–12.
ISSN 1613-0073.
Verfügbar unter: http://ceur-ws.org/Vol-1793/paper3.pdf
Zugriff am: 01. 08. 2018

Waschneck et al., 2016b

Waschneck, Bernd und Schmid, Gottfried, 2016b.
RAMI 4.0 in der Praxis: Vom Modell in den Reinraum.
IT & Production: Das Industrie 4.0-Magazin für erfolgreiche Produktion, **17** (5), S. 60–61.
ISSN 1439-7722.
Verfügbar unter: http://www.it-production.com/index.php?seite=einzel_artikel_ansicht&id=63587
Zugriff am: 01. 08. 2018

Waschneck et al., 2017

Waschneck, Bernd; Lee, Wei Fong Brian; Koh, Chey Woon Benny; Rippler, Christoph und Schmid, Gottfried, 2017.
Unified Frontend and Backend Industrie 4.0 Roadmap for Semiconductor Manufacturing.
In: *CEUR Workshop Proceedings: Science, Application and Methods in Industry 4.0 (SAMI) @i-KNOW*.
Graz, Oesterreich, 11. 10. 2017–12. 10. 2017, S. 2.1–20.
ISSN 1613-0073.
Verfügbar unter: `http://ceur-ws.org/Vol-2025/paper_sami40_2.pdf`
Zugriff am: 01. 08. 2018

Waschneck et al., 2018a

Waschneck, Bernd; Altenmüller, Thomas; Bauernhansl, Thomas und Kyek, Andreas, 2018a.
Case Study on Operator Compliance to Scheduling Decisions in Semiconductor Manufacturing.
In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*.
Muenchen, 20. 08. 2018–24. 08. 2018, S. 649–652.
ISBN 978-1-5386-3593-3.
DOI: `10.1109/COASE.2018.8560395`

Waschneck et al., 2018b

Waschneck, Bernd; Reichstaller, André; Belzner, Lenz; Altenmüller, Thomas; Bauernhansl, Thomas; Knapp, Alexander und Kyek, Andreas, 2018b.
Deep reinforcement learning for semiconductor production scheduling.
In: *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*.
Saratoga Springs, NY, USA, 30. 04. 2018–03. 05. 2018, S. 301–306.
ISSN 2376-6697.
DOI: 10.1109/ASMC.2018.8373191.

Ausgezeichnet mit dem Best Paper Award der Konferenz.

Waschneck et al., 2018c

Waschneck, Bernd; Reichstaller, André; Belzner, Lenz; Altenmüller, Thomas; Bauernhansl, Thomas; Knapp, Alexander und Kyek, Andreas, 2018c.
Optimization of global production scheduling with deep reinforcement learning.
Procedia CIRP, **72**, S. 1264–1269.
ISSN 2212-8271.
DOI: 10.1016/j.procir.2018.03.212.
Präsentiert bei: 51st CIRP Conference on Manufacturing Systems (CMS), Stockholm, Schweden, 18.05.2018-16.05.2018

Ausgezeichnet mit dem Best Paper Award der Konferenz bei 350 präsentierten Beiträgen.

Literaturverzeichnis

SEMI E10-0304(E) 2004
SEMI E10-0304(E) : Specification for Definition and Measurement of Equipment Reliability, Availability, and Maintainability (RAM), 2004.
San Jose, CA, USA, SEMI – Semiconductor Equipment und Materials International, Standard

SEMI E79-1106 2006
SEMI E79-1106 : Specification for Definition and Measurement of Equipment Productivity, 2006.
San Jose, CA, USA, SEMI – Semiconductor Equipment und Materials International, Standard

TR25 2008
TR25 : Technical Regulation 25: Manufacturing Equipment Performance: Reliability, Availability, Maintainability (RAM), 2008.
Neubiberg, Infineon Technologies AG, Interner Standard.
Nicht öffentlich zugänglich

TR27 2010
TR27 : Technical Regulation 27: Production Performance Indicators (PPI), 2010.
Neubiberg, Infineon Technologies AG, Interner Standard.
Nicht öffentlich zugänglich

International Technology Roadmap for Semiconductors 2011
International Technology Roadmap for Semiconductors : EXECUTIVE SUMMARY, 2011.
Semiconductor Industry Associations of the United States (SIA), Europe (ESIA), Japan (JEITA), South Korea (KSIA) and Taiwan (TSIA), Technologie Roadmap/Entwicklungsplanung.
Verfügbar unter: `http://www.itrs2.net/2011-itrs.html`

DIN 44300-9:1988-11 1988

DIN 44300-9:1988-11 : Informationsverarbeitung - Begriffe - Verarbeitunsabläufe, 1988.
Berlin, DIN Deutsches Institut für Normierung e.V.; Bezugsquelle: Beuth Verlag GmbH, Standard

Abadi et al., 2016

Abadi, Martín et al., 2016.
TensorFlow: A system for large-scale machine learning.
In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*.
Savannah, GA, USA, 02. 11. 2016–04. 11. 2016, S. 265–283.
ISBN 978-1-931971-33-1.
Verfügbar unter: `https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi`
Zugriff am: 01. 08. 2018

Amodei et al., 2018

Amodei, Dario und Hernandez, Danny, 2018.
AI and Compute.
San Francisco, CA, USA : OpenAI, US Type 501(c)(3) Nonprofit organization.
Verfügbar unter: `https://blog.openai.com/ai-and-compute/`
Zugriff am: 01. 08. 2018

Applied Materials, Inc., 2013

Applied Materials, Inc., 2013.
APPLIED APF RTD® AND REPORTER.
Santa Clara, CA, USA : Applied Global Services.
Verfügbar unter: `http://www.appliedmaterials.com/files/automation_software_resources/RTDReporterDatasheet.pdf`
Zugriff am: 01. 08. 2018

Arulkumaran et al., 2017

Arulkumaran, Kai; Deisenroth, Marc P.; Brundage, Miles und Bharath, Anil A., 2017.
Deep Reinforcement Learning: A Brief Survey.
IEEE Signal Processing Magazine, **34** (6), S. 26–38.
ISSN 1053-5888.
DOI: `10.1109/MSP.2017.2743240`

Aytug et al., 2005
Aytug, Haldun; Lawley, Mark A; McKay, Kenneth; Mohan, Shantha und Uzsoy, Reha, 2005.
Executing production schedules in the face of uncertainties: A review and some future directions.
European Journal of Operational Research, **161** (1), S. 86–110

Ballhaus et al., 2009
Ballhaus, Werner; Pagella, Alessandro und Vogel, Constantin, 2009.
Taktwechsel in der Halbleiterbranche?
Frankfurt am Main : PricewaterhouseCoopers AG Wirtschaftsprüfungsgesellschaft.
Verfügbar unter: `https://www.pwc.de/de/technologie-medien-und-telekommunikation/assets/taktwechsel-in-der-halbleiterbranche.pdf`
Zugriff am: 01. 08. 2018

Barr, 2007
Barr, Keith Elliott, 2007.
ASIC Design in the Silicon Sandbox : A Complete Guide to Building Mixed-Signal Integrated Circuits.
1. Aufl.
New York, NY, USA : McGraw-Hill.
ISBN 978-0071481618

Bauernhansl et al., 2014
Bauernhansl, Thomas; Ten Hompel, Michael und Vogel-Heuser, Birgit (Hrsg.), 2014.
Industrie 4.0 in Produktion, Automatisierung und Logistik : Anwendung, Technologien und Migration.
Wiesbaden : Springer Vieweg.
ISBN 978-3-658-04681-1.
DOI: `10.1007/978-3-658-04682-8`

Bauer et al., 2017

Bauer, Dennis; Maier, Florian; Bauernhansl, Thomas; Waschneck, Bernd; Ponsignon, Thomas; Gürster, Daniel; Oberegger, Bernhard; Felsberger, Andreas und Reiner, Gerald, 2017.
Concept and Possible Application of an Automated Framework to Influence Production Dispatch Based on Supply Chain Events.
In: Bousonville, Thomas; Melo, Teresa; Rezg, Nidhal und Vernadat, François (Hrsg.): *Proceedings of the 7th International Conference on Industrial Engineering and Systems Management (IESM 2017)*.
Saarbruecken, 11. 10. 2017–13. 10. 2017.
DOI: 10.5281/zenodo.1035179

Bauernhansl, 2003

Bauernhansl, Thomas, 2003.
Bewertung von Synergiepotenzialen im Maschinenbau.
Aachen, Lehrstuhl für Produktionssystematik des Laboratoriums für Werkzeugmaschinen und Betriebslehre (WZL) der RWTH Aachen, Deutscher Universitats-Verlag/GWV Fachverlage GmbH (Wiesbaden), Dissertation.
ISBN 978-3-8244-0688-3.
DOI: 10.1007/978-3-322-81082-3

Bauernhansl, 2014

Bauernhansl, Thomas, 2014.
Komplexität bewirtschaften - Einführung von Industrie 4.0 in Produktionssysteme.
MAV : Kompetenz in der spanenden Fertigung, **4**, S. 64–67

Bauernhansl, 2015

Bauernhansl, Thomas, 2015.
Vorlesung Fabrikbetriebslehre | VL07 Fertigungs- und Montagesystemplanung.
Stuttgart : Institut für Industrielle Fertigung und Fabrikbetrieb (IFF), Universität Stuttgart

Bischoff et al., 2015
Bischoff, Jürgen et al., 2015.
Erschließen der Potenziale der Anwendung von ‚Industrie 4.0' im Mittelstand: Kurzfassung der Studie.
Berlin / Muelheim an der Ruhr : Bundesministerium für Wirtschaft und Energie (BMWi) / agiplan GmbH.
Verfügbar unter: `http://www.zenit.de/fileadmin/Downloads/Studie_im_Auftrag_des_BMWi_Industrie_4.0_2015_agiplan_fraunhofer_iml_zenit_Kurzfassung.pdf`
Zugriff am: 01. 08. 2018

Blackstone et al., 1982
Blackstone, John H.; Phillips, Don T. und Hogg, Gary L., 1982.
A state-of-the-art survey of dispatching rules for manufacturing job shop operations.
The International Journal of Production Research, **20** (1), S. 27–45

Boström, 2014
Boström, Nick, 2014.
Superintelligence: Paths, Dangers, Strategies.
Oxford, UK : Oxford University Press.
ISBN 978-0199678112

Branke et al., 2015
Branke, Jürgen; Hildebrandt, Torsten und Scholz-Reiter, Bernd, 2015.
Hyper-heuristic evolution of dispatching rules: A comparison of rule representations.
Evolutionary computation, **23** (2), S. 249–277

Branke et al., 2016
Branke, Juergen; Nguyen, Su; Pickardt, Christoph W. und Zhang, Mengjie, 2016.
Automated design of production scheduling heuristics: A review.
IEEE Transactions on Evolutionary Computation, **20** (1), S. 110–124

Bradtke et al., 1994

Bradtke, Steven J. und Duff, Michael O., 1994.
Reinforcement Learning Methods for Continuous-time Markov Decision Problems.
In: *Proceedings of the 7th International Conference on Neural Information Processing Systems (NIPS'94)*.
Denver, CO, USA, 28. 11. 1994–01. 12. 1994, S. 393–400.
Verfügbar unter: http://dl.acm.org/citation.cfm?id=2998687.2998736
Zugriff am: 01. 08. 2018

Brauer et al., 1998

Brauer, Wilfried und Weiss, Gerhard, 1998.
Multi-machine scheduling – a multi-agent learning approach.
In: *Proceedings of the International Conference on Multi Agent Systems (Cat. No.98EX160)*.
Paris, Frankreich, 03. 07. 1998–07. 07. 1998, S. 42–48.
DOI: 10.1109/ICMAS.1998.699030

Brock et al., 2006

Brock, David C. und Moore, Gordon E., 2006.
Understanding Moore's law: four decades of innovation.
Philadelphia, PA, USA : Chemical Heritage Foundation.
ISBN 978-0-94190-141-3

Brockman et al., 2016

Brockman, Greg; Cheung, Vicki; Pettersson, Ludwig; Schneider, Jonas; Schulman, John; Tang, Jie und Zaremba, Wojciech, 2016.
OpenAI Gym.
Verfügbar unter: arXiv: 1606.01540
Zugriff am: 01. 08. 2018

Burke et al., 2010

Burke, Edmund K.; Hyde, Matthew; Kendall, Graham; Ochoa, Gabriela; Özcan, Ender und Woodward, John R., 2010.
A classification of hyper-heuristic approaches.
In: Gendreau, Michel und Potvin, Jean-Yves (Hrsg.): *Handbook of Metaheuristics*.
Boston, MA, USA : Springer US, S. 449–468.
ISBN 978-1-4419-1663-1.
DOI: 10.1007/978-1-4419-1665-5

Burton, 2017 Burton, Graeme, 2017.
TSMC says 3nm plant could cost it more than $20bn.
The Inquirer.
Verfügbar unter: https://www.theinquirer.net/inquirer/news/3018890/tsmc-says-3nm-plant-could-cost-it-more-than-usd20bn
Zugriff am: 01. 08. 2018

Buxmann et al., 2019
Buxmann, Peter und Schmidt, Holger, 2019.
Grundlagen der Künstlichen Intelligenz und des Maschinellen Lernens.
In: Buxmann, Peter und Schmidt, Holger (Hrsg.): *Künstliche Intelligenz: Mit Algorithmen zum wirtschaftlichen Erfolg*.
Berlin, Heidelberg : Springer-Verlag, S. 3–19.
ISBN 978-3-662-57568-0.
DOI: 10.1007/978-3-662-57568-0_1

Castellano, 2018
Castellano, Robert N., 2018.
Sizeable Changes In Semiconductor Equipment Market Share In 2017.
Seeking Alpha.
Verfügbar unter: https://seekingalpha.com/article/4143100-sizeable-changes-semiconductor-equipment-market-share-2017
Zugriff am: 01. 08. 2018

Cavalieri et al., 1999
Cavalieri, Sergio; Crisafulli, Francesco und Mirabella, Orazio, 1999.
A genetic algorithm for job-shop scheduling in a semiconductor manufacturing system.
In: *IECON'99. Conference Proceedings of the 25th Annual Conference of the IEEE Industrial Electronics Society (Cat. No.99CH37029)*.
San Jose, CA, USA, 29. 11. 1999–03. 12. 1999, S. 957–961

Chang et al., 1998
Chang, Chao-Fan und Chang, Shao-Kung, 1998.
A layer-based layout approach for semiconductor fabrication facilities.
In: *IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop (Cat. No.98CH36168)*.
Boston, MA, USA, 23. 09. 1998–25. 09. 1998, S. 385–390.
ISSN 1078-8743.
DOI: 10.1109/ASMC.1998.731626

Chan, 2016
Chan, Chih Ming, 2016.
Dynamic cycle-time improvement through big data analytics.
In: *2016 International Symposium on Semiconductor Manufacturing (ISSM)*.
Tokyo, Japan, 12. 12. 2016–13. 12. 2016, S. 1–4.
DOI: 10.1109/ISSM.2016.7934515

Chesbrough, 2007
Chesbrough, Henry W., 2007.
Why companies should have open business models.
MIT Sloan Management Review, **48** (2), S. 22

Chik et al., 2004
Chik, Mohd. Azizi; Ahmad, Ibrahim und Jamaluddin, Mohd. Yusoff, 2004.
An alternative approach to measures the application of dispatching rule in the wafer foundry.
In: *2004 IEEE International Conference on Semiconductor Electronics*.
Kuala Lumpur, Malaysia, 07. 12. 2004–09. 12. 2004, S. 642–644.
ISBN 0-7803-8658-2.
DOI: 10.1109/SMELEC.2004.1620968

Choi et al., 2011
Choi, Hyun-Seon; Kim, Ji-Su und Lee, Dong-Ho, 2011.
Real-time scheduling for reentrant hybrid flow shops: A decision tree based mechanism and its application to a TFT-LCD line.
Expert systems with Applications, **38** (4), S. 3514–3521

Chollet et al., 2015
Chollet, François et al., 2015.
Keras.
GitHub (online software development platform).
Verfügbar unter: https://keras.io
Zugriff am: 01. 08. 2018

Chollet, 2017 Chollet, François, 2017.
Deep Learning with Python.
Shelter Island, NY, USA : Manning Publications Co.
ISBN 978-1617294433

Chui et al., 2018 Chui, Michael; Manyika, James; Miremadi, Mehdi; Henke, Nicolaus; Chung, Rita; Nel, Pieter und Malhotra, Sankalp, 2018.
Notes from the AI frontier: Insights from hundreds of use cases.
McKinsey & Company, McKinsey Global Institute.
Verfügbar unter: https://www.mckinsey.de/~/media/McKinsey/Locations/Europe%20and%20Middle%20East/Deutschland/News/Presse/2018/2018-04-20/mgi_notes-from-ai-frontier_discussion-paper2.ashx
Zugriff am: 01. 08. 2018

Dewey, 2014 Dewey, Daniel, 2014.
Reinforcement learning and the reward engineering principle.
In: *2014 AAAI Spring Symposium Series*.
Stanford, CA, USA, 24. 03. 2014–26. 03. 2014.
Verfügbar unter: https://www.aaai.org/ocs/index.php/SSS/SSS14/paper/viewFile/7704/7740
Zugriff am: 01. 08. 2018

Doering et al., 2008
Doering, Robert und Nishi, Yoshio, 2008.
Handbook of Semiconductor Manufacturing Technology.
2. Aufl.
Boca Raton, FL, USA : CRC Press, Taylor & Francis Group.
ISBN 978-1574446753

Dudenredaktion, 2017
Dudenredaktion, 2017.
›automatisch‹ auf Duden online.
Berlin : Bibliographisches Institut GmbH.
Verfügbar unter: https://www.duden.de/node/646407/revisions/1635353/view
Zugriff am: 01. 08. 2018

Dudenredaktion, 2018a
Dudenredaktion, 2018a.
›autonom‹ auf Duden online.
Berlin : Bibliographisches Institut GmbH.
Verfügbar unter: https://www.duden.de/node/650163/revisions/1617396/view
Zugriff am: 01. 08. 2018

Dudenredaktion, 2018b
Dudenredaktion, 2018b.
›Compliance‹ auf Duden online.
Berlin : Bibliographisches Institut GmbH.
Verfügbar unter: https://www.duden.de/node/683850/revisions/1628922/view
Zugriff am: 01. 08. 2018

Eguchi et al., 1999
Eguchi, Toru; Oba, Fuminori und Hirai, Toshiki, 1999.
A neural network approach to dynamic job shop scheduling.
In: Mertins, K; Krause, O und Schallock, B (Hrsg.): *Global Production Management*.
Boston, MA, USA : Springer, Bd. 24, S. 152–159.
ISBN 978-1-4757-5334-9.
DOI: 10.1007/978-0-387-35569-6_19

Ehm et al., 2011
Ehm, Hans; Wenke, Hanna; Mönch, Lars; Ponsignon, Thomas und Forstner, Lisa, 2011.
Towards a supply chain simulation reference model for the semiconductor industry.
In: *Proceedings of the 2011 Winter Simulation Conference (WSC)*.
11. 12. 2011–14. 12. 2011, S. 2119–2130.
ISSN 1558-4305.
DOI: 10.1109/WSC.2011.6147925

Eisert, 2017 Eisert, Rebecca, 2017.
Halbleiterwert in der Autoindustrie: So entwickelt sich der Halbleitermarkt.
Automobilwoche: Die Branchen- und Wirtschaftszeitung. Crain Communications GmbH.
Verfügbar unter: https://www.automobilwoche.de/article/20171207/NACHRICHTEN/171209911/exklusiv-halbleiterwert-in-der-autoindustrie-so-entwickelt-sich-der-halbleitermarkt
Zugriff am: 01. 08. 2018

El-Bouri et al., 2000
El-Bouri, Ahmed; Balakrishnan, Subramaniam und Popplewell, Neil, 2000.
Sequencing jobs on a single machine: A neural network approach.
European Journal of Operational Research, **126** (3), S. 474–490.
ISSN 0377-2217.
DOI: 10.1016/S0377-2217(99)00302-1

Fordyce et al., 2015
Fordyce, Ken; Milne, R. John; Wang, Chi-Tai und Zisgen, Horst, 2015.
Modeling and Integration of Planning, Scheduling, and Equipment Configuration in Semiconductor Manufacturing: Part I. Review of Successes and Opportunities.
International Journal of Industrial Engineering: Theory, Applications and Practice, **22** (5), S. 575–600

François-Lavet et al., 2018
François-Lavet, Vincent; Henderson, Peter; Islam, Riashat; Bellemare, Marc G. und Pineau, Joelle, 2018.
An Introduction to Deep Reinforcement Learning.
Foundations and Trends in Machine Learning, **11** (3-4), S. 219–354.
ISSN 1935-8237.
DOI: 10.1561/2200000071

Friedli et al., 2018
Friedli, Thomas; Budde, Lukas; Benninghaus, Christoph; Elbe, Christian und Pejic, Tin, 2018.
Digital Technologies – Evolution of production in high-wage countries.
Institute of Technology Management, Universität St. Gallen, Benchmarking Report

Frosst et al., 2017

Frosst, Nicholas und Hinton, Geoffrey, 2017.
Distilling a Neural Network Into a Soft Decision Tree.
In: Besold, Tarek R. und Kutz, Oliver (Hrsg.): *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017)*.
Bari, Italien, 16. 11. 2017–17. 11. 2017.
ISSN 1613-0073.
Verfügbar unter: `http://ceur-ws.org/Vol-2071/CExAIIA_2017_paper_3.pdf`
Zugriff am: 01. 08. 2018

Gabel et al., 2007

Gabel, Thomas und Riedmiller, Martin, 2007.
Scaling adaptive agent-based reactive job-shop scheduling to large-scale problems.
In: *2007 IEEE Symposium on Computational Intelligence in Scheduling (SCIS'07)*.
Honolulu, HI, USA, 01. 04. 2007–05. 04. 2007, S. 259–266.
ISBN 1-4244-0704-4.
DOI: `10.1109/SCIS.2007.367699`

Géron, 2017 Géron, Aurélien, 2017.
Hands-On Machine Learning with Scikit-Learn and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems.
1. Aufl.
Sebastopol, CA, USA : O'Reilly Media, Inc.
ISBN 978-1491962299

Girau et al., 2013

Girau, Mike und Rose, Oliver, 2013.
Practical assessment of a combined dispatching policy at a high-mix low-volume asic facility.
In: *2013 Winter Simulations Conference (WSC)*.
Washington, DC, USA, 08. 12. 2013–11. 12. 2013, S. 3745–3755.
ISSN 0891-7736.
DOI: `10.1109/WSC.2013.6721734`

Goldstein, 1999

Goldstein, Jeffrey, 1999.
Emergence as a construct: History and issues.
Emergence, **1** (1), S. 49–72

Govind et al., 2008

Govind, Nirmal; Bullock, Eric W; He, Linling; Iyer, Bala; Krishna, Murali und Lockwood, Charles S, 2008.
Operations management in automated semiconductor manufacturing with integrated targeting, near real-time scheduling, and dispatching.
IEEE Transactions on Semiconductor Manufacturing, **21** (3), S. 363–370

Gu et al., 2016

Gu, Shixiang; Lillicrap, Timothy; Sutskever, Ilya und Levine, Sergey, 2016.
Continuous Deep Q-Learning with Model-based Acceleration.
In: Balcan, Maria F. und Weinberger, Kilian Q. (Hrsg.): *Proceedings of The 33rd International Conference on Machine Learning*.
New York, NY, USA, 20. 06. 2016–22. 06. 2016, S. 2829–2838.
Proceedings of Machine Learning Research.
Verfügbar unter: `http://proceedings.mlr.press/v48/gu16.pdf`
Zugriff am: 12. 02. 2019

Guidotti et al., 2018

Guidotti, Riccardo; Monreale, Anna; Turini, Franco; Pedreschi, Dino und Giannotti, Fosca, 2018.
A Survey Of Methods For Explaining Black Box Models.
arXiv preprint.
Verfügbar unter: `https://arxiv.org/abs/1802.01933`
Zugriff am: 01. 08. 2018

Gupta et al., 2006

Gupta, Amit Kumar und Sivakumar, Appa Iyer, 2006.
Job shop scheduling techniques in semiconductor manufacturing.
The International Journal of Advanced Manufacturing Technology, **27** (11-12), S. 1163–1169

Gurobi Optimization, Inc, 2015

Gurobi Optimization, Inc, 2015.
Gurobi optimizer reference manual.
Verfügbar unter: `http://www.gurobi.com`
Zugriff am: 01. 08. 2018

Gutenberg, 1951
Gutenberg, Erich, 1951.
Grundlagen der Betriebswirtschaftslehre : Die Produktion.
Berlin, Heidelberg : Springer-Verlag.
ISBN 978-3-662-21966-9

Ham et al., 2015 Ham, Andy Myoungsoo und Cho, Myeonsig, 2015.
A practical two-phase approach to scheduling of photolithography production.
IEEE Transactions on Semiconductor Manufacturing, **28** (3), S. 367–373.
ISSN 0894-6507.
DOI: 10.1109/TSM.2015.2451512

Hanny, 2011 Hanny, David, 2011.
Increase fab capacity: With predictive short-interval scheduling.
In: *2011 22nd IEEE/SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*.
Saratoga Springs, NY, USA, 16. 05. 2011–18. 05. 2011, S. 1–4.
ISBN 978-1-61284-408-4.
ISSN 2376-6697.
DOI: 10.1109/ASMC.2011.5898205

Hasselt et al., 2016
Hasselt, Hado van; Guez, Arthur und Silver, David, 2016.
Deep Reinforcement Learning with Double Q-Learning.
In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.
Phoenix, Arizona, USA, 12. 02. 2016–17. 02. 2016, S. 2094–2100.
AAAI'16.
Verfügbar unter: http://dl.acm.org/citation.cfm?id=3016100.3016191
Zugriff am: 01. 03. 2019

Hattori, 2015 Hattori, Takeshi, 2015.
From 20mm to 450mm: The Progress in Silicon Wafer Diameter Nodes.
Tokyo, Japan : Tokyo Electron Limited.
Verfügbar unter: https://www.tel.com/museum/magazine/material/150430_report04_03/
Zugriff am: 01. 08. 2018

Heck et al., 2011
Heck, Stefan; Kaza, Sri und Pinner, Dickon, 2011.
Creating value in the semiconductor industry.
McKinsey on Semiconductors, **1** (1), S. 5–14

Hessel et al., 2018
Hessel, Matteo et al., 2018.
Rainbow: Combining Improvements in Deep Reinforcement Learning.
In: *AAAI Publications, Thirty-Second AAAI Conference on Artificial Intelligence*.
New Orleans, NA, USA, 02. 02. 2018–07. 02. 2018.
Verfügbar unter: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17204
Zugriff am: 01. 08. 2018

Hester et al., 2018
Hester, Todd et al., 2018.
Deep Q-Learning from Demonstrations.
In: *Thirty-Second AAAI Conference on Artificial Intelligence*.
New Orleans, NA, USA, 02. 02. 2018–07. 02. 2018.
Verfügbar unter: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/download/16976/16682
Zugriff am: 12. 02. 2019.
Die Proceedings wurden am 29. April 2018 publiziert.

Hoffmann, 2017
Hoffmann, Daniela, 2017.
Infineon: Digitalisierung kann Standorte retten.
Produktion - Technik und Wirtschaft für die deutsche Industrie.
Verfügbar unter: https://www.produktion.de/veranstaltungen/fabrik-des-jahres/sieger/infineon-digitalisierung-kann-standorte-retten-257.html
Zugriff am: 01. 08. 2018

Holthaus et al., 1997
Holthaus, Oliver und Rajendran, Chandrasekharan, 1997.
Efficient dispatching rules for scheduling in a job shop.
International Journal of Production Economics, **48** (1), S. 87–105.
ISSN 0925-5273.
DOI: 10.1016/S0925-5273(96)00068-0

Holland, 2014
Holland, John H., 2014.
Complexity: A Very Short Introduction.
Oxford, UK : Oxford University Press.
ISBN 978-0199662548

Hopp et al., 2001
Hopp, Wallace J. und Spearman, Mark L., 2001.
Factory Physics: Foundations of Manufacturing Management.
2. Aufl.
New York, NY, USA : Irwin/McGraw-Hill.
ISBN 0-256-24795-1

Hruska, 2017 Hruska, Joel, 2017.
450mm silicon wafers aren't happening any time soon as major consortium collapses.
ExtremeTech.
Verfügbar unter: https://www.extremetech.com/computing/242699-450mm-silicon-wafers-arent-happening-time-soon-major-consortium-collapses
Zugriff am: 01. 08. 2018

Hsu, 2004 Hsu, Feng-Hsiung, 2004.
Behind Deep Blue: Building the Computer That Defeated the World Chess Champion.
Princeton, NJ, USA : Princeton University Press.
ISBN 978-0691118185

IHS Markit, 2017
IHS Markit, 2017.
Competitive Landscaping Tool - Q3 Update.
London, UK, IHS Markit Ltd, Technology Group, Marktanalyse

Ingimundardottir et al., 2011
Ingimundardottir, Helga und Runarsson, Thomas Philip, 2011.
Supervised Learning Linear Priority Dispatch Rules for Job-Shop Scheduling.
In: Coello, Carlos A. (Hrsg.): *International conference on Learning and Intelligent Optimization (LION)*.
Rom, Italien, 17. 01. 2011–21. 01. 2011, S. 263–277.
ISBN 978-3-642-25566-3.
DOI: 10.1007/978-3-642-25566-3_20

International Business Machines Corporation (IBM), 2018
International Business Machines Corporation (IBM), 2018.
IBM ILOG CPLEX Optimization Studio.
Verfügbar unter: https://www.ibm.com/de-de/marketplace/ibm-ilog-cplex
Zugriff am: 01. 08. 2018

Integrated Circuit Engineering Corp., 1997
Integrated Circuit Engineering Corp., 1997.
Changing Wafer Size and the Move to 300mm.
In: *Cost effective IC manufacturing*.
Scottsdale, AZ, USA : Integrated Circuit Engineering Corp.
ISBN 1877750603.
Verfügbar unter: http://smithsonianchips.si.edu/ice/cd/CEICM/SECTION7.pdf
Zugriff am: 01. 08. 2018

Jia et al., 2016 Jia, Zhao-hong; Wang, Chao und Leung, Joseph Y.-T., 2016.
An ACO algorithm for makespan minimization in parallel batch machines with non-identical job sizes and incompatible job families.
Applied Soft Computing, **38**, S. 395–404.
DOI: 10.1016/j.asoc.2015.09.056

Johnson, 2015 Johnson, R. Colin, 2015.
Samsung Breaks Ground on $14 Billion Fab.
EE Times, AspenCore Media.
Verfügbar unter: https://www.eetimes.com/document.asp?doc_id=1326565
Zugriff am: 01. 08. 2018

Jones, 2016 Jones, Handel, 2016.
FD SOI Benefits Rise at 14nm.
EE Times, AspenCore.
Verfügbar unter: https://www.eetimes.com/author.asp?section_id=36&doc_id=1329887
Zugriff am: 01. 08. 2018

Jouppi et al., 2017
Jouppi, Norman P. et al., 2017.
In-Datacenter Performance Analysis of a Tensor Processing Unit.
In: *Proceedings of the 44th Annual International Symposium on Computer Architecture*.
New York, NY, USA, 24. 06. 2017–28. 06. 2017, S. 1–12.
Proceedings of International Symposium on Computer Architecture.
ISBN 978-1-4503-4892-8.
DOI: 10.1145/3079856.3080246

Jovane et al., 2003
Jovane, Francesco; Koren, Yoram und Boer, Claudio R., 2003.
Present and Future of Flexible Automation: Towards New Paradigms.
CIRP Annals – Manufacturing Technology, **52** (2), S. 543–560.
DOI: 10.1016/S0007-8506(07)60203-0

Kahn, 2018
Kahn, Jeremy, 2018.
Sky-High Salaries Are the Weapons in the AI Talent War.
Bloomberg Businessweek.
Verfügbar unter: https://www.bloomberg.com/news/articles/2018-02-13/in-the-war-for-ai-talent-sky-high-salaries-are-the-weapons
Zugriff am: 01. 08. 2018

Kallenberg, 2002
Kallenberg, Lodewijk, 2002.
Finite State and Action MDPS : Methods and Applications.
In: Feinberg, Eugene A. und Shwartz, Adam (Hrsg.): *Handbook of Markov Decision Processes*.
New York, NY, USA : Springer Science & Business Media, Bd. 40.
International Series in Operations Research & Management Science.
ISBN 978-0-7923-7459-6.
ISSN 0884-8289.
DOI: 10.1007/978-1-4615-0805-2

Keil, 2012
Keil, Sophia, 2012.
Flussorientierte Gestaltung von Produktionssystemen : Anwendung am Beispiel von Halbleiterfabriken.
Dresden, Lehrstuhl für Betriebswirtschaftslehre der Technischen Universität Dresden, Gabler Verlag (Wiesbaden), Dissertation.
ISBN 978-3-8349-3996-8.
DOI: 10.1007/978-3-8349-3997-5

Kempf, 1994
Kempf, Karl, 1994.
Intel five-machine six step mini-fab description.
Intel/ASU Report, Arizona State University.
Verfügbar unter: http://aar.faculty.asu.edu/research/intel/papers/fabspec.html
Zugriff am: 01. 08. 2018

Kern et al., 2015

Kern, Wolfgang; Rusitschka, Fabian; Kopytynski, Witold; Eckl, S und Bauernhansl, Thomas, 2015.
Alternatives to assembly line production in the automotive industry.
In: *The 23rd International Conference on Production Research*.
Manila, Philippinen, 01. 08. 2015–04. 08. 2015, S. 1–9

Khan et al., 2018

Khan, Hassan N.; Hounshell, David A. und Fuchs, Erica R.H., 2018.
Science and research policy at the end of Moore's law.
Nature Electronics, **1** (1), S. 14

Kim et al., 2017 Kim, Namyong und Shin, Hayong, 2017.
The application of actor-critic reinforcement learning for fab dispatching scheduling.
In: *2017 Winter Simulation Conference (WSC)*.
Las Vegas, NV, USA, 03. 12. 2017–06. 12. 2017, S. 4570–4571.
ISSN 1558-4305.
DOI: 10.1109/WSC.2017.8248209

Kingman et al., 2003

Kingman, John F.C. und Atiyah, Michael F., 2003.
The single server queue in heavy traffic.
Operations management: critical perspectives on business and management, **57**, S. 40

Kindratenko et al., 2009

Kindratenko, Volodymyr V.; Enos, Jeremy J.; Shi, Guochun; Showerman, Michael T.; Arnold, Galen W.; Stone, John E.; Phillips, James C. und Hwu, Wen-mei, 2009.
GPU clusters for high-performance computing.
In: *2009 IEEE International Conference on Cluster Computing and Workshops*.
New Orleans, LA, USA, 31. 08. 2009–04. 09. 2009, S. 1–8.
ISSN 1552-5244.
DOI: 10.1109/CLUSTR.2009.5289128

Kingma et al., 2015
Kingma, Diederik P. und Ba, Jimmy, 2015.
Adam: A method for stochastic optimization.
In: *Contribution to International Conference on Learning Representations (ICLR)*.
San Diego, CA, USA, 07. 05. 2015–09. 05. 2015.
Verfügbar unter: http://hdl.handle.net/11245/1.505367
Zugriff am: 01. 08. 2018

Klemmt et al., 2009
Klemmt, Andreas; Weigert, Gerald; Almeder, Christian und Mönch, Lars, 2009.
A comparison of MIP-based decomposition techniques and VNS approaches for batch scheduling problems.
In: *Proceedings of the 2009 Winter Simulation Conference (WSC)*.
Austin, TX, USA, 13. 12. 2009–16. 12. 2009, S. 1686–1694.
ISBN 978-1-4244-5770-0.
ISSN 0891-7736.
DOI: 10.1109/WSC.2009.5429173

Klemmt, 2012
Klemmt, Andreas, 2012.
Ablaufplanung in der Halbleiter-und Elektronikproduktion : Hybride Optimierungsverfahren und Dekompositionstechniken.
Dresden, Fakultät Elektrotechnik und Informationstechnik der Technischen Universität Dresden, Vieweg+Teubner Verlag (Wiesbaden), Dissertation.
ISBN 978-3-8348-1993-2.
DOI: 10.1007/978-3-8348-1994-9

Kornmeier, 2007
Kornmeier, Martin, 2007.
Wissenschaftstheorie und wissenschaftliches Arbeiten: eine Einführung für Wirtschaftswissenschaftler.
Heidelberg : Physica-Verlag.
ISBN 978-3-7908-1918-2

Koza et al., 1996 Koza, John R.; Bennett, Forrest H.; Andre, David und Keane, Martin A., 1996.
Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming.
In: Gero, John S. und Sudweeks, Fay (Hrsg.): *Artificial Intelligence in Design '96*.
Dordrecht : Springer Netherlands, S. 151–170.
ISBN 978-94-009-0279-4.
DOI: 10.1007/978-94-009-0279-4_9

Ku et al., 2016 Ku, Wen-Yang und Beck, J. Christopher, 2016.
Mixed Integer Programming models for job shop scheduling: a computational analysis.
Computers & Operations Research, **73**, S. 165–173.
DOI: 10.1016/j.cor.2016.04.006

Kubicek, 1976 Kubicek, Herbert, 1976.
Heuristische Bezugsrahmen und heuristisch angelegte Forschungsdesign als Elemente einer Konstruktionsstrategie empirischer Forschung.
In: Köhler, Richard (Hrsg.) (Hrsg.): *Empirische und handlungstheoretische Forschungskonzeption in der Betriebswirtschaftslehre*,
S. 3–36.
ISBN 3-7910-0214-7

Kuhn, 1996 Kuhn, Thomas S., 1996.
The Structure of Scientific Revolutions.
3. Aufl.
Chicago, IL, USA : University of Chicago Press.
ISBN 978-0226458083

Lakatos, 1982 Lakatos, Imre, 1982.
Einleitung: Wissenschaft und Pseudowissenschaft.
In: *Die Methodologie der wissenschaftlichen Forschungsprogramme*.
Wiesbaden : Vieweg+Teubner Verlag, S. 1–6.
ISBN 978-3-663-08082-4.
DOI: 10.1007/978-3-663-08082-4_1

LeCun et al., 2015 LeCun, Yann; Bengio, Yoshua und Hinton, Geoffrey, 2015.
Deep learning.
Nature, **521** (7553), S. 436.
DOI: 10.1038/nature14539

Li et al., 2009 Li, Li; Qiao, Fei und Wu, Qidi, 2009.
ACO-based multi-objective scheduling of parallel batch processing machines with advanced process control constraints.
The International Journal of Advanced Manufacturing Technology, **44** (9-10), S. 985–994.
DOI: 10.1007/s00170-008-1904-8

Lillicrap et al., 2016
Lillicrap, Timothy P.; Hunt, Jonathan J.; Pritzel, Alexander; Heess, Nicolas; Erez, Tom; Tassa, Yuval; Silver, David und Wierstra, Daan, 2016.
Continuous control with deep reinforcement learning.
Verfügbar unter: https://arxiv.org/pdf/1509.02971.pdf
Zugriff am: 01. 03. 2019

Lin et al., 2013 Lin, Chi-Wei; Lin, Yang-Kuei und Hsieh, Han-Ting, 2013.
Ant colony optimization for unrelated parallel machine scheduling.
The International Journal of Advanced Manufacturing Technology, **67** (1-4), S. 35–45.
DOI: 10.1007/s00170-013-4766-7

Lipton, 2018 Lipton, Zachary C., 2018.
The Mythos of Model Interpretability.
Queue, **16** (3), S. 30:31–30:57.
ISSN 1542-7730.
DOI: 10.1145/3236386.3241340

Little, 1961 Little, John D. C., 1961.
A Proof for the Queuing Formula: L = λW.
Operations Research, **9** (3), S. 383–387.
DOI: 10.1287/opre.9.3.383

Lödding, 2016 Lödding, Hermann, 2016.
Verfahren der Fertigungssteuerung : Grundlagen, Beschreibung, Konfiguration.
3. Aufl.
Berlin, Heidelberg : Springer Vieweg.
ISBN 978-3-662-48458-6.
DOI: 10.1007/978-3-662-48459-3

Luo et al., 2018 Luo, Yixin; Ghose, Saugata; Cai, Yu; Haratsch, Erich F. und Mutlu, Onur, 2018.
HeatWatch: Improving 3D NAND Flash Memory Device Reliability by Exploiting Self-Recovery and Temperature Awareness.
In: *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*.
Wien, Oesterreich, 24. 02. 2018–28. 02. 2018, S. 504–517.
ISSN 2378-203X.
DOI: 10.1109/HPCA.2018.00050

Maaten et al., 2008
Maaten, Laurens van der und Hinton, Geoffrey, 2008.
Visualizing data using t-SNE.
Journal of Machine Learning Research, **9** (Nov), S. 2579–2605.
ISSN 1533-7928.
Verfügbar unter: http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf
Zugriff am: 01. 08. 2018

Mahadevan et al., 1997
Mahadevan, Sridhar; Marchalleck, Nicholas; Das, Tapas K. und Gosavi, Abhijit, 1997.
Self-Improving Factory Simulation using Continuous-time Average-Reward Reinforcement Learning.
In: Fisher, Douglas H. (Hrsg.): *Proceedings of the 14th International Conference on Machine Learning (ICML '97)*.
Nashville, TN, USA, 08. 07. 1997–12. 07. 1997, S. 202–210.
ISBN 1-55860-486-3

Mahadevan et al., 1998
Mahadevan, Sridhar und Theocharous, Georgios, 1998.
Optimizing Production Manufacturing Using Reinforcement Learning.
In: Cook, Diane J. (Hrsg.): *Proceedings of the Eleventh International Florida Artificial Intelligence Research Society (FLAIRS) Conference*.
Sanibel Island, FL, USA, 18. 05. 2011–20. 05. 2011, S. 372–377.
ISBN 1-57735-051-0

Mao et al., 2016 Mao, Hongzi; Alizadeh, Mohammad; Menache, Ishai und Kandula, Srikanth, 2016.
Resource management with deep reinforcement learning.
In: *Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets '16)*.
Atlanta, GA, USA, 09. 11. 2016–10. 11. 2016, S. 50–56.
ISBN 978-1-4503-4661-0.
DOI: 10.1145/3005745.3005750

Mason et al., 2002
Mason, Scott J.; Fowler, John W. und Carlyle, W. Matthew, 2002.
A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops.
Journal of Scheduling, **5** (3), S. 247–262.
DOI: 10.1002/jos.102

MATLAB, 2015 MATLAB, 2015.
Version R2015b.
Natick, Massachusetts : The MathWorks Inc.
Verfügbar unter: https://de.mathworks.com/help/releases/R2015b/matlab/index.html
Zugriff am: 01. 08. 2018

McClean et al., 2018
McClean, Bill und IC Insights, Inc., 2018.
The McClean Report 2018 : A Complete Analysis and Forecast of the Integrated Circuit Industry.
Scottsdale, AZ, USA : IC Insights, Inc.
Verfügbar unter: http://www.icinsights.com/services/mcclean-report/
Zugriff am: 01. 08. 2018.
Technischer Bericht/ Marktanalyse

Meindl et al., 2012
Meindl, Bernhard und Templ, Matthias, 2012.
Deliverable D3, Subtask 10: Analysis of commercial and free and open source solvers for linear optimization problems.
Eurostat and Statistics Netherlands within the ESSnet project on common tools and harmonised methodology for SDC in the ESS.
Verfügbar unter: https://pdfs.semanticscholar.org/7bd9/116754b4d5a9a7ac2e08ae169e9e168a4eab.pdf
Zugriff am: 01. 08. 2018

Metz, 2016 Metz, Cade, 2016.
In two moves, AlphaGo and Lee Sedol redefined the future.
Wired (Computerzeitschrift), Condé Nast Verlag.
Verfügbar unter: https://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/
Zugriff am: 01. 08. 2018

Metz, 2018 Metz, Cade, 2018.
Mark Zuckerberg, Elon Musk and the Feud Over Killer Robots.
The New York Times.
Verfügbar unter: https://www.nytimes.com/2018/06/09/technology/elon-musk-mark-zuckerberg-artificial-intelligence.html
Zugriff am: 01. 08. 2018

Mnih et al., 2013
Mnih, Volodymyr; Kavukcuoglu, Koray; Silver, David; Graves, Alex; Antonoglou, Ioannis; Wierstra, Daan und Riedmiller, Martin, 2013.
Playing Atari with Deep Reinforcement Learning.
In: *Presented at the conference on Neural Information Processing Systems (NIPS), Deep Learning Workshop 2013*.
Lake Tahoe, NV, USA, 05. 12. 2013–10. 12. 2013.
Verfügbar unter: https://arxiv.org/abs/1312.5602
Zugriff am: 01. 08. 2018

Mnih et al., 2015
Mnih, Volodymyr et al., 2015.
Human-level control through deep reinforcement learning.
Nature, **518** (7540), S. 529.
DOI: 10.1038/nature14236

Mnih et al., 2016
Mnih, Volodymyr; Badia, Adria P.; Mirza, Mehdi; Graves, Alex; Lillicrap, Timothy; Harley, Tim; Silver, David und Kavukcuoglu, Koray, 2016.
Asynchronous Methods for Deep Reinforcement Learning.
In: Balcan, Maria F. und Weinberger, Kilian Q. (Hrsg.): *Proceedings of The 33rd International Conference on Machine Learning*.
New York, NY, USA, 20. 06. 2016–22. 06. 2016, S. 1928–1937.
Proceedings of Machine Learning Research.
Verfügbar unter: `http://proceedings.mlr.press/v48/mniha16.pdf`
Zugriff am: 12. 02. 2019

Möhrle et al., 2018a
Möhrle, Martin G. und Specht, Dieter, 2018a.
Gabler Wirtschaftslexikon, Stichwort: Market Pull.
Wiesbaden : Springer Gabler Verlag (Herausgeber).
Verfügbar unter: `https://wirtschaftslexikon.gabler.de/definition/market-pull-37331/version-260768`
Zugriff am: 01. 08. 2018

Möhrle et al., 2018b
Möhrle, Martin G. und Specht, Dieter, 2018b.
Gabler Wirtschaftslexikon, Stichwort: S-Kurven-Konzept.
Wiesbaden : Springer Gabler Verlag (Herausgeber).
Verfügbar unter: `http://wirtschaftslexikon.gabler.de/definition/s-kurven-konzept-43411/version-266741`
Zugriff am: 01. 08. 2018

Möhrle, 2018
Möhrle, Martin G., 2018.
Gabler Wirtschaftslexikon, Stichwort: Technology Push.
Wiesbaden : Springer Gabler Verlag (Herausgeber).
Verfügbar unter: `https://wirtschaftslexikon.gabler.de/definition/technology-push-50304/version-273524`
Zugriff am: 01. 08. 2018

Mönch et al., 2003

Mönch, Lars; Stehli, Marcel und Zimmermann, Jens, 2003.
FABMAS: An Agent-Based System for Production Control of Semiconductor Manufacturing Processes.
In: Mařík, Vladimír; McFarlane, Duncan und Valckenaers, Paul (Hrsg.): *Holonic and Multi-Agent Systems for Manufacturing*.
Berlin, Heidelberg : Springer-Verlag, S. 258–267.
ISBN 978-3-540-45185-3

Mönch et al., 2006a

Mönch, Lars; Stehli, Marcel; Zimmermann, Jens und Habenicht, Ilka, 2006a.
The FABMAS multi-agent-system prototype for production control of water fabs: design, implementation and performance assessment.
Production Planning & Control, **17** (7), S. 701–716.
DOI: 10.1080/09537280600901269

Mönch et al., 2006b

Mönch, Lars; Zimmermann, Jens und Otto, Peter, 2006b.
Machine learning techniques for scheduling jobs with incompatible families and unequal ready times on parallel batch machines.
Engineering Applications of Artificial Intelligence, **19** (3), S. 235–245.
DOI: 10.1016/j.engappai.2005.10.001

Mönch et al., 2011

Mönch, Lars; Fowler, John W.; Dauzère-Pérès, Stéphane; Mason, Scott J. und Rose, Oliver, 2011.
A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations.
Journal of Scheduling, **14** (6), S. 583–599.
ISSN 1099-1425.
DOI: 10.1007/s10951-010-0222-9

Mönch et al., 2013

Mönch, Lars; Fowler, John W. und Mason, Scott J., 2013.
Production Planning and Control for Semiconductor Wafer Fabrication Facilities : Modeling, Analysis, and Systems.
In: Sharda, Ramesh und Vo, Stefan (Hrsg.):
Operations Research/Computer Science Interfaces Series.
ISBN 978-1-4614-4471-8.
DOI: 10.1007/978-1-4614-4472-5

Moore, 1965 Moore, Gordon E., 1965.
Cramming more components onto integrated circuits.
Electronics, **38** (8), S. 114–117

Müller et al., 2016
Müller, Felix Georg; Bressner, Markus; Görzig, David; Röber, Thomas und Hrsg.: Bauernhansl, Thomas, 2016.
Industrie 4.0: Entwicklungsfelder für den Mittelstand : Aktuelle Hemmnisse und konkreter Bedarf.
Stuttgart : Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA.
Verfügbar unter: `http://publica.fraunhofer.de/dokumente/N-410653.html`
Zugriff am: 01. 08. 2018

Muth et al., 1963
Muth, John F.; Thompson, Gerald Luther und Winters, Peter R. (Hrsg.), 1963.
Industrial scheduling.
Englewood Cliffs, NJ, USA : Prentice-Hall.
Prentice-Hall International Series in Management.
Verfügbar unter: `https://trove.nla.gov.au/work/21364061`
Zugriff am: 01. 08. 2018

Nyhuis et al., 2012
Nyhuis, Peter und Wiendahl, Hans-Peter, 2012.
Logistische Kennlinien: Grundlagen, Werkzeuge und Anwendungen.
3. Aufl.
Berlin, Heidelberg : Springer-Verlag.
ISBN 978-3-540-92838-6

Ouelhadj et al., 2009
Ouelhadj, Djamila und Petrovic, Sanja, 2009.
A survey of dynamic scheduling in manufacturing systems.
Journal of scheduling, **12** (4), S. 417–431.
DOI: `10.1007/s10951-008-0090-8`

Ovacik et al., 1997
Ovacik, Irfan M. und Uzsoy, Reha, 1997.
Decomposition Methods for Complex Factory Scheduling Problems.
New York, NY, USA : Springer Science+Business Media New York.
ISBN 978-0-7923-9835-6.
DOI: `10.1007/978-1-4615-6329-7`

Ozawa et al., 1999
Ozawa, Katsutoshi; Wada, Hideyuki und Yamaguchi, Tsuyoshi, 1999.
Optimum tool planning using the X-factor theory.
In: *1999 IEEE International Symposium on Semiconductor Manufacturing Conference Proceedings (Cat No.99CH36314)*.
Santa Clara, CA, USA, 11. 10. 1999–13. 10. 1999, S. 49–52.
ISSN 1523-553X.
DOI: 10.1109/ISSM.1999.808735

Paternina-Arboleda et al., 2005
Paternina-Arboleda, Carlos D. und Das, Tapas K., 2005.
A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problem.
Simulation Modelling Practice and Theory, **13** (5), S. 389–406.
DOI: 10.1016/j.simpat.2004.12.003

Pennachin et al., 2007
Pennachin, Cassio und Goertzel, Ben, 2007.
Contemporary Approaches to Artificial General Intelligence.
In: Goertzel, Ben und Pennachin, Cassio (Hrsg.): *Artificial General Intelligence*.
Berlin, Heidelberg : Springer-Verlag, S. 1–30.
ISBN 978-3-540-68677-4.
DOI: 10.1007/978-3-540-68677-4_1

Pfeffer, 2012
Pfeffer, Markus, 2012.
Simulationsgestützte Untersuchung von logistischen Optimierungsstrategien bei Halbleiterfertigungsprozessen.
Erlangen, Technische Fakultät der Universität Erlangen-Nürnberg, Dissertation.
https://d-nb.info/1022002287/34

Pickardt et al., 2013
Pickardt, Christoph W.; Hildebrandt, Torsten; Branke, Jürgen; Heger, Jens und Scholz-Reiter, Bernd, 2013.
Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems.
International Journal of Production Economics, **145** (1), S. 67–77.
DOI: 10.1016/j.ijpe.2012.10.016

Pinedo, 2016 Pinedo, Michael L., 2016.
Scheduling: theory, algorithms, and systems.
5. Aufl.
Berlin : Springer.
ISBN 978-3-319-26578-0.
DOI: 10.1007/978-3-319-26580-3

Plappert, 2016 Plappert, Matthias, 2016.
keras-rl.
GitHub (online software development platform).
Verfügbar unter: https://github.com/matthiasplappert/keras-rl
Zugriff am: 01. 08. 2018

Popper et al., 2013
Popper, Karl R. und Hrsg.: Keuth, Herbert, 2013.
Logik der Forschung.
Berlin : Walter de Gruyter.
ISBN 978-3050057088

Popper, 1973 Popper, Karl R., 1973.
Objektive Erkenntnis: ein evolutionärer Entwurf.
Hamburg : Hoffmann und Campe.
Hoffmann u. Campe. Kritische Wissenschaft.
ISBN 978-3455090888

Porter, 1998 Porter, Michael E., 1998.
Competitive Advantage : Creating and Sustaining Superior Performance: With a new Introduction.
New York, NY, USA : The Free Press, a division of Simon & Schuster Inc.
ISBN 0684841460.
1. Auflage wurde 1985 publiziert.

Pratt et al., 1996
Pratt, Lorien und Jennings, Barbara, 1996.
A Survey of Transfer Between Connectionist Networks.
Connection Science, **8** (2), S. 163–184.
DOI: 10.1080/095400996116866

Puffer, 2007 Puffer, Walter, 2007.
Technisch-ökonomische Effizienzbetrachtungen für die Halbleiterfertigung.
Muenchen, Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München, Dissertation.
https://mediatum.ub.tum.de/doc/606194/606194.pdf

Puget, 2016 Puget, Jean François, 2016.
The Most Popular Language For Machine Learning Is ....
Online Blog auf IBM developerWorks®.
Verfügbar unter: https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Language_Is_Best_For_Machine_Learning_And_Data_Science
Zugriff am: 01. 08. 2018

Qu et al., 2016 Qu, Shuhui; Wang, Jie; Govil, Shivani und Leckie, James O., 2016.
Optimized Adaptive Scheduling of a Manufacturing Process System with Multi-skill Workforce and Multiple Machine Types: An Ontology-based, Multi-agent Reinforcement Learning Approach.
In: *Factories of the Future in the digital environment - Proceedings of the 49th CIRP Conference on Manufacturing Systems*.
Stuttgart, 25. 05. 2016–27. 05. 2016, S. 55–60.
DOI: 10.1016/j.procir.2016.11.011

Ribeiro et al., 2016
Ribeiro, Marco Tulio; Singh, Sameer und Guestrin, Carlos, 2016.
›Why Should I Trust You?‹ Explaining the Predictions of Any Classifier.
In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
San Francisco, CA, USA, 13. 08. 2016–17. 08. 2016, S. 1135–1144.
ISBN 978-1-4503-4232-2.
DOI: 10.1145/2939672.2939778

Riedmiller et al., 1999
Riedmiller, Simone und Riedmiller, Martin, 1999.
A neural reinforcement learning approach to learn local dispatching policies in production scheduling.
In: *International Joint Conferences on Artificial Intelligence Organization (IJCAI)*.
Stockholm, Schweden, 31. 07. 1999–06. 08. 1999, S. 764–771.
Verfügbar unter: https://www.ijcai.org/Proceedings/99-2/Papers/016.pdf
Zugriff am: 01. 08. 2018

Rosen et al., 2015
Rosen, Roland; Wichert, Georg von; Lo, George und Bettenhausen, Kurt D, 2015.
About the importance of autonomy and digital twins for the future of manufacturing.
IFAC-PapersOnLine, **48** (3), S. 567–572.
DOI: 10.1016/j.ifacol.2015.06.141

Rummery et al., 1994
Rummery, Gavin A. und Niranjan, Mahesan, 1994.
On-line Q-learning using connectionist systems.
University of Cambridge, Department of Engineering Cambridge, England.
Verfügbar unter: http://mi.eng.cam.ac.uk/reports/svr-ftp/auto-pdf/rummery_tr166.pdf
Zugriff am: 01. 03. 2019.
Technische Mitteilung

Russell et al., 2010
Russell, Stuart J. und Norvig, Peter, 2010.
Artificial Intelligence : A Modern Approach.
3. Aufl.
Harlow, UK : Pearson Education Limited.
ISBN 978-0-13-604259-4

Samuel, 1959
Samuel, Arthur Lee, 1959.
Some Studies in Machine Learning Using the Game of Checkers.
IBM Journal of Research and Development, **3** (3), S. 210–229.
ISSN 0018-8646.
DOI: 10.1147/rd.33.0210

Schubert et al., 2014
Schubert, Enrico; Heyder, Jan; Bauer, Florian; Waschneck, Bernd; Stumpf, Wolfgang; Wegscheider, Werner; Delft, Jan von; Ludwig, Stefan und Högele, Alexander, 2014.
Toward combined transport and optical studies of the 0.7-anomaly in a quantum point contact.
physica status solidi (b), **251** (9), S. 1931–1937.
DOI: 10.1002/pssb.201350218

Schuh et al., 2014
Schuh, Günther und Schmidt, Carsten, 2014.
Produktionsmanagement : Handbuch Produktion und Management 5.
Berlin, Heidelberg : Springer-Verlag.
VDI-Buch.
ISBN 978-3-642-54287-9.
DOI: 10.1007/978-3-642-54288-6

Schulman et al., 2015
Schulman, John; Levine, Sergey; Abbeel, Pieter; Jordan, Michael und Moritz, Philipp, 2015.
Trust Region Policy Optimization.
In: Bach, Francis und Blei, David (Hrsg.): *Proceedings of the 32nd International Conference on Machine Learning*.
Lille, Frankreich, 07. 07. 2015–09. 07. 2015, S. 1889–1897.
Proceedings of Machine Learning Research.
Verfügbar unter: http://proceedings.mlr.press/v37/schulman15.pdf
Zugriff am: 12. 02. 2019

Schaul et al., 2016
Schaul, Tom; Quan, John; Antonoglou, Ioannis und Silver, David, 2016.
Prioritized Experience Replay.
In: *International Conference on Learning Representations 2016*.
San Juan, Puerto Rico, 02. 05. 2016–04. 05. 2016.
ICLR'16.
Verfügbar unter: https://arxiv.org/pdf/1511.05952.pdf
Zugriff am: 01. 03. 2019

Schulman et al., 2016
Schulman, John; Moritz, Philipp; Levine, Sergey; Jordan, Michael und Abbeel, Pieter, 2016.
High-Dimensional Continuous Control Using Generalized Advantage Estimation.
In: *International Conference on Learning Representations 2016*.
San Juan, Puerto Rico, 02. 05. 2016–04. 05. 2016.
ICLR'16.
Verfügbar unter: https://arxiv.org/pdf/1506.02438.pdf
Zugriff am: 01. 03. 2019

Schulman et al., 2017
Schulman, John; Wolski, Filip; Dhariwal, Prafulla; Radford, Alec und Klimov, Oleg, 2017.
Proximal policy optimization algorithms.
arXiv preprint.
Verfügbar unter: https://arxiv.org/abs/1707.06347
Zugriff am: 01. 03. 2019

Scholz, 2005
Scholz, Peter, 2005.
Softwareentwicklung eingebetteter Systeme : Grundlagen, Modellierung, Qualitätssicherung.
Berlin, Heidelberg : Springer-Verlag.
Xpert.press.
ISBN 978-3-540-23405-0.
DOI: 10.1007/3-540-27522-3

Shanthikumar et al., 2007
Shanthikumar, J. George; Ding, Shengwei und Zhang, Mike Tao, 2007.
Queueing theory for semiconductor manufacturing systems: a survey and open problems.
IEEE Transactions on Automation Science and Engineering, **4** (4), S. 513–522.
DOI: 10.1109/TASE.2007.906348

Shoham et al., 2009
Shoham, Yoav und Leyton-Brown, Kevin, 2009.
Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations.
New York, NY, USA : Cambridge University Press.
ISBN 978-0521899437

Silver et al., 2014
Silver, David; Lever, Guy; Heess, Nicolas; Degris, Thomas; Wierstra, Daan und Riedmiller, Martin, 2014.
Deterministic Policy Gradient Algorithms.
In: Xing, Eric P. und Jebara, Tony (Hrsg.): *Proceedings of the 31st International Conference on Machine Learning*.
Peking, China, 22. 06. 2014–24. 06. 2014, S. 387–395.
Proceedings of Machine Learning Research, Nr. 1.
Verfügbar unter: http://proceedings.mlr.press/v32/silver14.pdf
Zugriff am: 01. 03. 2019

Silver et al., 2016
Silver, David et al., 2016.
Mastering the game of Go with deep neural networks and tree search.
Nature, **529** (7587), S. 484–489.
DOI: 10.1038/nature16961

Silver et al., 2017a
Silver, David et al., 2017a.
Mastering the game of go without human knowledge.
Nature, **550** (7676), S. 354.
DOI: 10.1038/nature24270

Silver et al., 2017b
Silver, David et al., 2017b.
Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.
arXiv preprint.
Verfügbar unter: https://arxiv.org/abs/1712.01815
Zugriff am: 01. 08. 2018

Silver, 2015
Silver, David, 2015.
UCL Course on Reinforcement Learning.
London, UK : University College London (UCL), Department of Computer Science. Vorlesungsskript.
Verfügbar unter: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
Zugriff am: 01. 08. 2018

Stricker et al., 2018
Stricker, Nicole; Kuhnle, Andreas; Sturm, Roland und Friess, Simon, 2018.
Reinforcement learning for adaptive order dispatching in the semiconductor industry.
CIRP Annals, **67** (1), S. 511–514.
ISSN 0007-8506.
DOI: 10.1016/j.cirp.2018.04.041

Sutton et al., 1998
Sutton, Richard S. und Barto, Andrew G., 1998.
Reinforcement learning: An introduction.
2. Aufl.
Cambridge, MA, USA : MIT Press.
ISBN 978-0-262-19398-6

Talkhestani et al., 2017

Talkhestani, Behrang Ashtari; Schlögl, Wolfgang und Weyrich, Michael, 2017.
Synchronisierung von digitalen Modellen : Anwendung einer Ankerpunktmethode für Fertigungszellen.
atp magazin, **59** (07-08), S. 62–69.
ISSN 2364-3137.
DOI: 10.17560/atp.v59i07-08.1902

Talkhestani et al., 2018

Talkhestani, Behrang Ashtari; Jazdi, Nasser; Schlögl, Wolfgang und Weyrich, Michael, 2018.
A concept in synchronization of virtual production system with real factory based on anchor-point method.
ISSN 2212-8271.
DOI: 10.1016/j.procir.2017.12.168.
Präsentiert bei: 11th CIRP Conference on Intelligent Computation in Manufacturing Engineering, Golf von Neapel, Italien, 19.07.2017–21.07.2017

Tichatschke, 2008

Tichatschke, Rainer, 2008.
"Auf den Schultern von Giganten" : Zur Geschichte der Mathematischen Optimierung.
Mathematik/Informatik, Universität Trier, Forschungsbericht Nr. 08-4.
Verfügbar unter: https://www.uni-trier.de/fileadmin/fb4/INF/TechReports/History-article.pdf

Tieleman et al., 2012

Tieleman, Tijmen und Hinton, Geoffrey, 2012.
Lecture 6.5 – RMSProp: Divide the gradient by a running average of its recent magnitude.
COURSERA: Neural networks for machine learning, **4** (2), S. 26–31.
Verfügbar unter: https://www.coursera.org/lecture/neural-networks/rmsprop-divide-the-gradient-by-a-running-average-of-its-recent-magnitude-YQHki
Zugriff am: 01. 08. 2018

Turing, 2009 Turing, Alan M., 2009.
Computing Machinery and Intelligence.
In: Epstein, Robert; Roberts, Gary und Beber, Grace (Hrsg.): *Parsing the Turing Test: Computing Machinery and IntelligencePhilosophical and Methodological Issues in the Quest for the Thinking Computer*.
1. Aufl.
Dordrecht, Niederlanden : Springer Netherlands, S. 23–65.
ISBN 978-1-4020-6708-2.
DOI: 10.1007/978-1-4020-6710-5

Ulrich et al., 2001
Ulrich, Hans und Schwaninger, Markus (Hrsg.), 2001.
Systemorientiertes Management : Das Werk von Hans Ulrich.
Bern, Schweiz : Haupt.
ISBN 978-3-258-06359-1

Ulrich et al., 1976a
Ulrich, Peter und Hill, Wilhelm, 1976a.
Wissenschaftstheoretische Grundlagen der Betriebswirtschaftslehre (Teil 1).
Wirtschaftswissenschaftliches Studium: Zeitschrift für Ausbildung und Hochschulkontakt, **5** (7), S. 304–309

Ulrich et al., 1976b
Ulrich, Peter und Hill, Wilhelm, 1976b.
Wissenschaftstheoretische Grundlagen der Betriebswirtschaftslehre (Teil 2).
Wirtschaftswissenschaftliches Studium: Zeitschrift für Ausbildung und Hochschulkontakt, **5** (8), S. 345–350

Ulrich, 1970 Ulrich, Hans, 1970.
Die Unternehmung als produktives soziales System: Grundlagen der allgemeinen Unternehmungslehre.
Bern – Stuttgart : Verlag Paul Haupt.
Unternehmung und Unternehmungsführung

Uzsoy et al., 1993
Uzsoy, Reha; Church, Laura K.; Ovacik, Irfan M. und Hinchman, Jim, 1993.
Performance evaluation of dispatching rules for semiconductor testing operations.
Journal of Electronics Manufacturing, **3** (02), S. 95–105.
DOI: 10.1142/S0960313193000115

Voigt et al., 2017
Voigt, Kai-Ingo und Steven, Marion, 2017.
Gabler Wirtschaftslexikon, Stichwort: Produktionstypen.
Wiesbaden : Springer Gabler Verlag (Herausgeber).
Verfügbar unter: http://wirtschaftslexikon.gabler.de/Archiv/13463/produktionstypen-v8.html
Zugriff am: 01. 08. 2018

Voigt, 2010
Voigt, Kai-Ingo, 2010.
Gabler Wirtschaftslexikon, Stichwort: Werkstattproduktion.
Wiesbaden : Springer Gabler Verlag (Herausgeber).
Verfügbar unter: http://wirtschaftslexikon.gabler.de/Archiv/55465/werkstattproduktion-v5.html
Zugriff am: 01. 08. 2018

Voigt, 2015
Voigt, Kai-Ingo, 2015.
Gabler Wirtschaftslexikon, Stichwort: Automatisierung.
Wiesbaden : Springer Gabler Verlag (Herausgeber).
Verfügbar unter: http://wirtschaftslexikon.gabler.de/Archiv/72569/automatisierung-v7.html
Zugriff am: 01. 08. 2018

Wang et al., 2002
Wang, Cheng-Shuo und Uzsoy, Reha, 2002.
A genetic algorithm to minimize maximum lateness on a batch processing machine.
Computers & Operations Research, **29** (12), S. 1621–1640.
DOI: 10.1016/S0305-0548(01)00031-4

Wang et al., 2016
Wang, Ziyu; Schaul, Tom; Hessel, Matteo; Hasselt, Hado; Lanctot, Marc und Freitas, Nando, 2016.
Dueling Network Architectures for Deep Reinforcement Learning.
In: Balcan, Maria F. und Weinberger, Kilian Q. (Hrsg.): *Proceedings of The 33rd International Conference on Machine Learning*.
New York, NY, USA, 20. 06. 2016–22. 06. 2016, S. 1995–2003.
Proceedings of Machine Learning Research.
Verfügbar unter: http://proceedings.mlr.press/v48/wangf16.pdf
Zugriff am: 01. 03. 2019

Waschneck et al., 2016a
Waschneck, Bernd; Altenmüller, Thomas; Bauernhansl, Thomas und Kyek, Andreas, 2016a.
Production Scheduling in Complex Job Shops from an Industry 4.0 Perspective: A Review and Challenges in the Semiconductor Industry.
In: *CEUR Workshop Proceedings: Science, Application and Methods in Industry 4.0 (SAMI) @i-KNOW*.
Graz, Oesterreich, 19. 10. 2016, S. 3.1–12.
ISSN 1613-0073.
Verfügbar unter: `http://ceur-ws.org/Vol-1793/paper3.pdf`
Zugriff am: 01. 08. 2018

Waschneck et al., 2016b
Waschneck, Bernd und Schmid, Gottfried, 2016b.
RAMI 4.0 in der Praxis: Vom Modell in den Reinraum.
IT & Production: Das Industrie 4.0-Magazin für erfolgreiche Produktion, **17** (5), S. 60–61.
ISSN 1439-7722.
Verfügbar unter: `http://www.it-production.com/index.php?seite=einzel_artikel_ansicht&id=63587`
Zugriff am: 01. 08. 2018

Waschneck et al., 2017
Waschneck, Bernd; Lee, Wei Fong Brian; Koh, Chey Woon Benny; Rippler, Christoph und Schmid, Gottfried, 2017.
Unified Frontend and Backend Industrie 4.0 Roadmap for Semiconductor Manufacturing.
In: *CEUR Workshop Proceedings: Science, Application and Methods in Industry 4.0 (SAMI) @i-KNOW*.
Graz, Oesterreich, 11. 10. 2017–12. 10. 2017, S. 2.1–20.
ISSN 1613-0073.
Verfügbar unter: `http://ceur-ws.org/Vol-2025/paper_sami40_2.pdf`
Zugriff am: 01. 08. 2018

Waschneck et al., 2018a

Waschneck, Bernd; Altenmüller, Thomas; Bauernhansl, Thomas und Kyek, Andreas, 2018a.
Case Study on Operator Compliance to Scheduling Decisions in Semiconductor Manufacturing.
In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*.
Muenchen, 20. 08. 2018–24. 08. 2018, S. 649–652.
ISBN 978-1-5386-3593-3.
DOI: 10.1109/COASE.2018.8560395

Waschneck et al., 2018b

Waschneck, Bernd; Reichstaller, André; Belzner, Lenz; Altenmüller, Thomas; Bauernhansl, Thomas; Knapp, Alexander und Kyek, Andreas, 2018b.
Deep reinforcement learning for semiconductor production scheduling.
In: *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*.
Saratoga Springs, NY, USA, 30. 04. 2018–03. 05. 2018, S. 301–306.
ISSN 2376-6697.
DOI: 10.1109/ASMC.2018.8373191.

Ausgezeichnet mit dem Best Paper Award der Konferenz.

Waschneck et al., 2018c

Waschneck, Bernd; Reichstaller, André; Belzner, Lenz; Altenmüller, Thomas; Bauernhansl, Thomas; Knapp, Alexander und Kyek, Andreas, 2018c.
Optimization of global production scheduling with deep reinforcement learning.
Procedia CIRP, **72**, S. 1264–1269.
ISSN 2212-8271.
DOI: 10.1016/j.procir.2018.03.212.
Präsentiert bei: 51st CIRP Conference on Manufacturing Systems (CMS), Stockholm, Schweden, 18.05.2018-16.05.2018

Ausgezeichnet mit dem Best Paper Award der Konferenz bei 350 präsentierten Beiträgen.

Watkins et al., 1992
Watkins, Christopher J. C. H. und Dayan, Peter, 1992.
Q-learning.
Machine learning, **8** (3-4), S. 279–292.
ISSN 0885-6125.
DOI: 10.1007/BF00992698

Weckman et al., 2008
Weckman, Gary R.; Ganduri, Chandrasekhar V. und Koonce, David A., 2008.
A neural network job-shop scheduler.
Journal of Intelligent Manufacturing, **19** (2), S. 191–201.
ISSN 0956-5515.
DOI: 10.1007/s10845-008-0073-9

Weiss, 1999
Weiss, Mitchell, 1999.
New twists on 300 mm fab design and layout.
Semiconductor International, **22** (8), S. 103

Whitt, 1983
Whitt, Ward, 1983.
The queueing network analyzer.
Bell Labs Technical Journal, **62** (9), S. 2779–2815

Wichert, 2000
Wichert, Andreas, 2000.
Künstliche Intelligenz.
Verfügbar unter: http://www.spektrum.de/lexikon/neurowissenschaft/kuenstliche-intelligenz/6810
Zugriff am: 01. 08. 2018

Wiendahl, 2014
Wiendahl, Hans-Peter, 2014.
Betriebsorganisation für Ingenieure.
8. Aufl.
Muenchen : Carl Hanser Verlag GmbH & Co. KG.
ISBN 978-3-446-44053-1

Xpress, 2018
Xpress, Fair Isaac Corporation (FICO), 2018.
FICO Xpress Optimization.
Verfügbar unter: http://www.fico.com/de/products/fico-xpress-optimization
Zugriff am: 01. 08. 2018

Yoon et al., 2008

Yoon, Hyun Joong und Shen, Weiming, 2008.
A multiagent-based decision-making system for semiconductor wafer fabrication with hard temporal constraints.
IEEE Transactions on Semiconductor Manufacturing, **21** (1), S. 83–91.
DOI: 10.1109/TSM.2007.914388

Zahavy et al., 2016

Zahavy, Tom; Ben-Zrihem, Nir und Mannor, Shie, 2016.
Graying the black box: Understanding DQNs.
In: *ICML'16 Proceedings of the 33rd International Conference on International Conference on Machine Learning*.
New York, NY, USA, 19. 06. 2016–24. 06. 2016, S. 1899–1908

Zhang et al., 2017

Zhang, Tao; Xie, Shufang und Rose, Oliver, 2017.
Real-time job shop scheduling based on simulation and Markov decision processes.
In: *2017 Winter Simulation Conference (WSC)*.
Las Vegas, NV, USA, 03. 12. 2017–06. 12. 2017, S. 3899–3907.
ISSN 1558-4305.
DOI: 10.1109/WSC.2017.8248100

Zhang et al., 1995

Zhang, Wei und Dietterich, Thomas G., 1995.
A Reinforcement Learning Approach to Job-shop Scheduling.
In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence – Volume 2*.
Montreal, Quebec, Canada, 20. 08. 1995–25. 08. 1995, S. 1114–1120.
Proceedings of the International Joint Conference on Artificial Intelligence.
ISBN 1-55860-363-8

Die Variabilität in der kundenindividuellen Massenproduktion stellt eine enorme Herausforderung für die industrielle Fertigung dar. Die komplexe Werkstattfertigung als Produktionsprinzip eignet sich aufgrund der inhärenten Flexibilität besonders für die kundenindividuelle Massenproduktion. Allerdings sind die bestehenden Methodiken für die Produktionssteuerung einer Werkstattfertigung für die Einmal- oder Wiederholproduktion ausgelegt.

In der vorliegenden Arbeit wird eine Methodik für eine dezentrale, selbstorganisierte und autonome Produktionssteuerung für eine Werkstattfertigung entwickelt, die dazu beiträgt, mit der zunehmenden Komplexität und dem gesteigerten Produktionsvolumen umzugehen. Dabei wird die Produktion als Reinforcement-Learning-Modell formalisiert, in dem mehrere kooperative Deep-Q-Network-Agenten lernen, die Abarbeitungsreihenfolge zu optimieren.

Die Erprobung der Methodik in zwei praxisnahen Fallbeispielen aus der Halbleiterindustrie zeigt ihre Leistungsfähigkeit. In beiden Fallbeispielen konnten Strategien zur Optimierung der Abarbeitungsreihenfolge auf oder über Expertenniveau autonom erlernt werden.

ISBN 978-3-8396-1623-9

9 783839 616239

FRAUNHOFER VERLAG