Bei diesem Beitrag handelt es sich um einen wissenschaftlich begutachteten und freigegebenen Fachaufsatz („reviewed paper“).

doi.org/10.37544/1436-4980-2025-11-12-90
Datum der Einreichung: 06.10.2025
Datum der Annahme: 04.11.2025
Datum der Veröffentlichung: 17.12.2025

Automatisierung von Planungs- und Steuerungsaufgaben in der Produktion

Reinforcement Learning in der Produktionssteuerung

E. Uhlmann, J. Polte, C. Mühlich

ZUSAMMENFASSUNG Die Steuerung von Produktionsprozessen stellt aufgrund der hohen Prozesskomplexität eine zentrale Herausforderung für produzierende Unternehmen dar. Vor diesem Hintergrund gewinnt die Automatisierung von Planungs- und Steuerungsaufgaben durch Algorithmen zunehmend an Bedeutung. Methoden des Reinforcement Learning bieten vielversprechendes Potenzial, um diese Herausforderung zu adressieren. Dieser Beitrag vergleicht Methoden des Reinforcement Learning mit exakten und metaheuristischen Algorithmen, um die Einsatzgrenzen und die Konkurrenzfähigkeit lernbasierter Verfahren im aktuellen Entwicklungsstand zu bewerten.

STICHWÖRTER
Maschinenbelegungsplanung, Reinforcement Learning, Automatisierung

Reinforcement Learning for production control

ABSTRACT Due to the high complexity of processes, production control represents a key challenge for manufacturing companies. In this regard, the automation of planning and control tasks using algorithms is becoming increasingly important. Reinforcement Learning methods offer promising potential to address these challenges. This article compares Reinforcement Learning methods with exact and metaheuristic algorithms in order to evaluate the limits of application and competitiveness of learning-based methods in their current state of development.

1 Einleitung

Bei der Herstellung von Produkten können vielfältige Faktoren den Produktionsablauf stören und die Erreichbarkeit von Produktionszielgrößen wie die Liefertermintreue oder die Ressourcenauslastung gefährden. Aufgabe der Produktionssteuerung ist es, trotz dieser Störeinflüsse, die effiziente Bearbeitung von Produktionsaufträgen zu ermöglichen [1]. Im Hinblick darauf stellt die Organisation des Produktionsablaufs für produzierende Unternehmen eine zentrale Herausforderung dar. Neben unerwarteten Ereignissen wie Personal- und Maschinenausfällen müssen unter anderem Material-, Ressourcenverfügbarkeiten, Liefertermine und Instandhaltungsbedarfe berücksichtigt werden, die sich stets verändern können. Insbesondere Unternehmen mit einer hohen Variantenvielfalt und geringen Losgrößen müssen unter solchen Bedingungen ihre Produktionsabläufe kontinuierlich überwachen und an veränderte Bedingungen flexibel anpassen.

Vor diesem Hintergrund hat das Bestreben, Planungs- und Steuerungsaufgaben mithilfe von Algorithmen zu automatisieren, langjährigen Bestand in der angewandten Forschung. Im Forschungsfeld Operations Research wurden in der Vergangenheit zahlreiche Ansätze für die Lösung von Maschinenbelegungsproblemen präsentiert, die sich allgemein in exakte und approximative Verfahren einteilen lassen [2]. Dabei gewinnen insbesondere lernbasierte Verfahren angesichts der technologischen Fortschritte im Bereich der Halbleitertechnik und der Künstlichen Intelligenz zunehmend an Bedeutung, da sie die Bestimmung komplexer Entscheidungsstrategien ermöglichen. Für das Lernern optimaler Entscheidungsstrategien im Kontext der Produktionssteuerung weisen Methoden des Reinforcement Learning (RL) ein vielversprechendes Potenzial auf [3]. Im RL lernen Agenten aus der Interaktion mit einer simulierten Produktionsumgebung, wie zur Erreichung verschiedenartiger Produktionsziele Aufträge auf Ressourcen ideal verteilt werden können. Dabei agieren Agenten nach dem Fehler-Irrtum-Prinzip, indem sie die Wirkung unbekannter Entscheidungsstrategien durch Exploration in einer simulierten Produktionsumgebung bestimmen und die Erkenntnisse in die Steuerung zukünftiger Verhaltensweisen einfließen lassen. Maßgebend dabei ist die sogenannte Belohnungsfunktion, die einem Agenten signalisiert, inwiefern eine Entscheidung zur Erreichung vordefinierter Zielgrößen beiträgt. Positive Lernsignale bestärken einen Agenten in der jeweiligen Verhaltensweise. Negative Lernsignale führen hingegen dazu, dass ein Agent entsprechende Entscheidungen als Handlungsoption abwertet. Aufgabe eines Agenten ist es, das individuelle Entscheidungsverhalten so anzupassen, dass die positiven Lernsignale maximiert und das zugehörige Optimierungsziel erreicht wird.

Methoden des RL zeichnen sich durch die Fähigkeit aus, generalisierbare Entscheidungsstrategien für komplexe und hochdynamische Produktionssysteme zu approximieren, während exakte Verfahren infolge des erhöhten Rechenaufwands in ihrer Skalierbarkeit eingeschränkt sind. Trotz des Potenzials von RL-Metho-

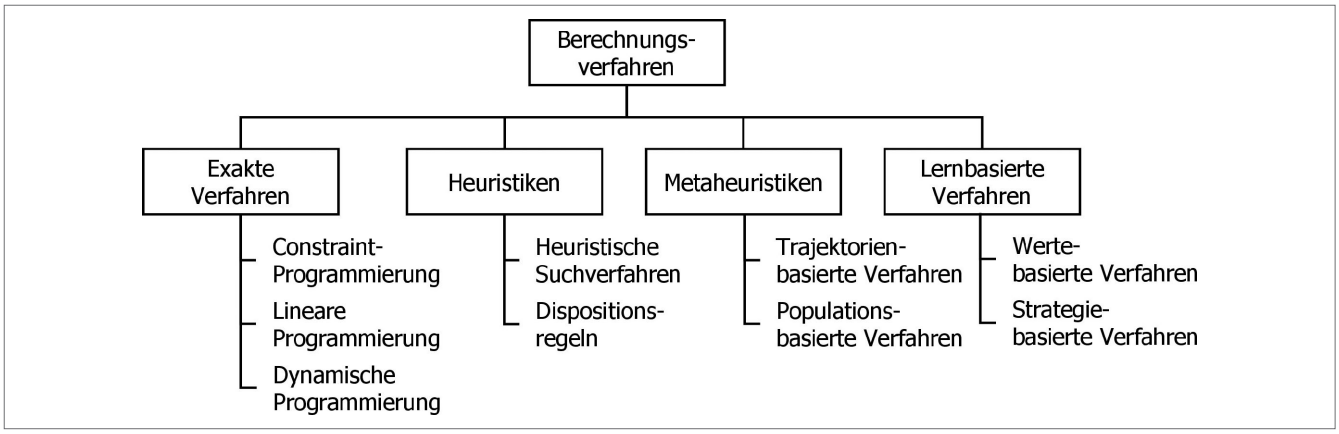

Bild 1 Einteilung von Algorithmen zur Lösung von Problemen der Maschinenbelegungsplanung. *Grafik: Fraunhofer IPK*

den liegen empirische Nachweise für deren industriellen Nutzen lediglich in begrenztem Umfang vor, wodurch Aussagen über die industrielle Anwendbarkeit nur bedingt möglich sind. Um die Einsatzgrenzen von RL zu bestimmen, benötigt es einen systematischen Vergleich zwischen lernbasierten und konventionellen Berechnungsverfahren. Ziel des Beitrags ist es, durch die Gegenüberstellung von Berechnungsverfahren die zentralen Unterscheidungsmerkmale der Verfahrenskategorien zu identifizieren und zu charakterisieren. Anhand ausgewählter Benchmark-Datensätze werden die Eigenschaften sowie Vor- und Nachteile der jeweiligen Berechnungsverfahren bestimmt, die produzierende Unternehmen zur Bewertung des Potenzials von Algorithmen für den Einsatz in der Produktionssteuerung nutzen können.

2 Stand der Technik

Algorithmen zur Lösung von Maschinenbelegungsproblemen lassen sich wie in **Bild 1** dargestellt in exakte, heuristische, metaheuristische sowie lernbasierte Verfahren einteilen. Exakte Berechnungsverfahren zeichnen sich dadurch aus, dass diese das Finden der optimalen Lösung garantieren, sofern ausreichend Zeit für die Berechnung gegeben ist. In dieser Verfahrenskategorie werden Planungsprobleme zumeist als gemischt-ganzzahliges lineares Optimierungsproblem oder als Optimierungsproblem mit Nebenbedingungen formuliert. Beide Modellparadigmen nutzen Baumsuchverfahren wie den Branch-and-Bound-Algorithmus [4], um den Lösungsraum systematisch zu durchsuchen und die Bestimmung einer optimalen Lösung mathematisch abzusichern. Zentrale Unterschiede bestehen jedoch in der Problemmodellierung sowie im Ablauf und Mechanismus der Suchverfahren. Ein wesentlicher Nachteil der exakten Berechnungsverfahren ist die begrenzte Skalierbarkeit. Die Maschinenbelegung beschreibt in der industriellen Praxis aufgrund der Vielzahl von Aufträgen, Ressourcen und variablen Einflussfaktoren ein NP-schweres Planungsproblem, das weder trivial in Nebenbedingungen und ganzzahligen Variablen zu modellieren noch in akzeptabler Zeit optimal zu lösen ist. In der Folge beschränkt sich der Einsatz auf kleine bis mittlere Probleminstanzen, wobei die Weiterentwicklung der Suchverfahren in den vergangenen Jahren die Leistungskapazitäten deutlich erhöht haben. Namhafte Vertreter umfassen kommerzielle Solver wie Cplex [5], Gurobi [6] und CP-SAT von Google OR Tools [7]. Während Cplex und Gurobi ein gemischt-ganzzahliges lineares Optimierungsproblem lösen, kombiniert CP-SAT die Constraint-Programmierung mit verschiedenen Techniken wie die boolsche Erfüllbarkeitssuche. Damit kann CP-SAT effizient zulässige Lösungen für komplexe Planungsaufgaben bestimmen und diese mit fortschreitender Rechenzeit iterativ verbessern.

Während sich exakte Berechnungsverfahren aufgrund des exponentiell steigenden Rechenaufwands nur bedingt für praxisnahe Planungsaufgaben eignen, können Heuristiken und Metaheuristiken Lösungen für komplexe Problemstellungen in akzeptabler Zeit approximieren. Klassische Heuristiken beruhen dabei insbesondere auf problemspezifischen Dispositionsregeln wie First-In-First-Out oder die Priorisierung nach der kürzesten Bearbeitungszeit [8]. Heuristiken dieser Art können interdisziplinäres Domänenwissen integrieren und Lösungen in kurzen Berechnungszeiten generieren, die jedoch in der Lösungsgüte alternativen Berechnungsverfahren zumeist unterlegen sind. Metaheuristiken hingegen kombinieren verschiedene Heuristiken in ein gemeinsames Suchverfahren, um Optimierungsprobleme unabhängig von der Problemstruktur effizient zu lösen. Diese Verfahren lassen sich grundsätzlich in populations- und trajektorienbasierte Verfahren einteilen, wobei der zentrale Unterschied in der Anzahl der betrachteten Lösungskandidaten besteht [9]. Populationsbasierte Verfahren beginnen die Lösungssuche mit einer Menge von Kandidaten, die mithilfe naturinspirierter Mechanismen wie Selektion und Mutation iterativ weiterentwickelt werden. Der bekannteste Vertreter dieser Kategorie ist der genetische Algorithmus, der in vielfachen Variationen und Kombinationen mit alternativen Verfahren zur Lösung von Planungsproblemen eingesetzt wird. Trajektorienbasierte Metaheuristiken betrachten dementgegen nur einen Lösungskandidaten, der unter Verwendung lokaler Suchverfahren in seiner Lösungsgüte inkrementell optimiert wird. Relevante Algorithmen umfassen die Tabu-Suche, Simulated Annealing oder die variable Nachbarschaftssuche. Eine umfassende Übersicht zu heuristischen und metaheuristischen Ansätzen im Kontext der Maschinenbelegungsplanung ist in den Arbeiten von *Chaudry* et al. [10] sowie *Türkyilmaz* et al. [11] zu finden.

Analog zu heuristischen und metaheuristischen Verfahren lassen sich mithilfe lernbasierter Ansätze Lösungen für komplexe Planungsprobleme approximieren. Eine besondere Rolle im Kontext der Forschung spielen dabei Methoden des RL, die aufgrund ihrer Kombination mit Deep Learning Methoden Planungsprobleme unterschiedlicher Schwierigkeitsgrade näherungsweise lösen können. Im RL lernt ein Agent durch die Interaktion mit einer simulierten Umgebung optimale Entscheidungen zu treffen. Der

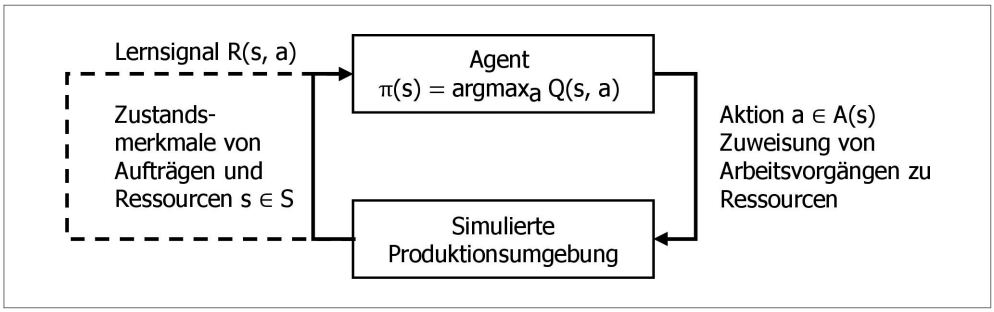

Bild 2 Simulationsbasierte Interaktionen eines Agenten im Kontext von RL. *Grafik: in Anlehnung an Sutton und Barto [14]*

Lernprozess erfolgt iterativ, indem ein Agent wiederholt Entscheidungen trifft, diese ausführt und anschließend durch den Erhalt einer Belohnung oder Bestrafung eine Rückmeldung über die Güte der getroffenen Entscheidung erhält. Ziel eines Agenten ist es, durch die fortlaufende Optimierung der verfolgten Handlungsstrategie die Belohnung über die Simulationszeit hinweg zu maximieren. Obwohl das Training von RL-Modellen in der Regel mit hohem Aufwand verbunden ist, können gut trainierte Modelle vergleichbare Lösungen in kurzen Rechenzeiten wie heuristische Verfahren erzielen [12]. Algorithmen wie Deep-Q-Networks (DQN) und Actor-Critic gehören zu den fortschrittlichsten Ansätzen zur Lösung komplexer Entscheidungsstrategien, die aufgrund ihres Potenzials für den Einsatz in der Produktionssteuerung zunehmend in den Fokus wissenschaftlicher Untersuchungen rücken [13, 14].

3 Reinforcement Learning

3.1 Markov Entscheidungsprozess

Methoden des RL lösen formal das Problem eines Markov-Entscheidungsprozesses (MEP) [14]. **Bild 2** zeigt die Grundstruktur eines MEP, in dem ein Agent durch die Auswahl und Ausführung einer Entscheidung mit einer Umgebung in Interaktion tritt. Eine Entscheidungssituation wird durch den Zustand s bestimmt, in dem sich ein Agent zum Zeitpunkt t befindet. Der Zustand s enthält jene Merkmale, die den Zustand einer Produktionsumgebung charakterisieren und relevant für die Bestimmung eines Belegungsplans sind. Dies umfasst Auftragsmerkmale wie Prozesszeiten und Maschinenalternativen pro Arbeitsvorgang sowie auch die Merkmale der am Produktionsprozess beteiligten Ressourcen. Zu Letzteren gehören insbesondere leistungsbestimmende Merkmale wie die Auslastung der Arbeitssysteme.

In jedem Zustand s wählt ein Agent eine Aktion a auf Grundlage seiner Entscheidungsstrategie $\pi(s)$ aus. Der Aktionsraum $A(s)$ bezeichnet dabei die Menge aller Aktionen, die von einem Agenten in einem Zustand s ausführbar sind. Im Fall der Produktionssteuerung kann der Aktionsraum $A(s)$ durch die Menge aller zulässigen Zuweisungsmöglichkeiten von Arbeitsvorgängen zu Produktionsressourcen beschrieben werden. Nach Ausführung einer Aktion a in der simulierten Produktionsumgebung erhält ein Agent ein Lernsignal durch die Belohnungsfunktion $R(s, a)$, welche die Güte einer getroffenen Entscheidung als skalaren Wert r ausdrückt. Je höher der Belohnungswert r, desto größer der Beitrag der getroffenen Entscheidung zur Zielerreichung.

Im RL nutzt ein Agent das Lernsignal der simulierten Produktionsumgebung, um die verfolgte Entscheidungsstrategie $\pi(s)$ inkrementell zu verbessern. Die Entscheidungsstrategie $\pi(s)$ kann mithilfe der sogenannten Q-Funktion $Q(s, a)$ bestimmt werden. Die Q-Funktion besagt, welche kumulative Belohnung ein Agent über den Verlauf der Simulation erwarten kann, wenn er eine Aktion a im Zustand s ausführt. Damit kann ein Agent für einen gegebenen Zustand s die Entscheidung mit dem maximalen Erwartungswert ermitteln, indem er die Q-Werte aller verfügbaren Entscheidungsoptionen vergleicht und diejenige mit dem Höchstwert auswählt. Die Bestimmung einer optimalen Q-Funktion $Q^*(s, a)$ beschreibt die Lösung eines MEP. Diese kann in wertebasierten RL-Methoden sowohl in tabellarischer Form mithilfe stochastischer Aktualisierungsregeln als auch mit Funktionsapproximatoren wie tiefen neuronalen Netzen näherungsweise ermittelt werden.

3.2 Modell der Produktionsumgebung

In diesem Beitrag erfolgt das Training der RL-Modelle in einer simulierten Produktionsumgebung, in der ausschließlich Produktionsaufträge und Maschinen als planungsrelevante Entitäten betrachtet werden. Ein Produktionsauftrag besteht aus einer Menge von Arbeitsvorgängen, die entsprechend einer technisch bedingten Bearbeitungsreihenfolge auszuführen sind. Jeder Arbeitsvorgang kann durch eine Maschine oder mehrere Maschinen mit unterschiedlichen Prozesszeiten bearbeitet werden. Im Beitrag wird die Aufgabe der Produktionssteuerung als Mehrzieloptimierungsproblem betrachtet. Ziel der Produktionssteuerung ist es, die mittleren Auftragsdurchlaufzeiten ZDL_m zu minimieren und zugleich die Leistungsdifferenzen aller Produktionsressourcen auf einem möglichst hohem Auslastungsniveau A_m auszugleichen. Die Belohnungsfunktion $R(s, a)$ setzt sich demzufolge gemäß Formel 1 aus einem durchlaufzeit- und einem leistungsorientierten Anteil zusammen. Beide Belohnungsterme werden als normierte Größen berechnet, wobei höhere Werte einem größeren Zielerfüllungsgrad entsprechen.

$$R(s, a) = w_{ZDL}\, R_{ZDL} + (1 - w_{ZDL})\, R_A \qquad (1)$$

Der durchlaufzeitorientierte Anteil R_{ZDL} formuliert gemäß Formel 2 das Ziel der Durchlaufzeitminimierung als Lernsignal, indem Belegungspläne mit einer mittleren Auftragsdurchlaufzeit ZDL_m nahe dem theoretischen Minimum ZDL_{min} die höchsten Belohnungswerte erhalten.

$$R_{ZDL} = 1 - \frac{ZDL_{rel} - 1}{ZDL_{rel,max} - 1} \qquad (2)$$

Hierzu wird das relative Durchlaufzeitverhältnis ZDL_{rel} nach Formel 3 als Quotient der tatsächlichen und minimalen Auftragsdurchlaufzeit ausgedrückt. Je größer das relative Durchlaufzeitverhältnis ZDL_{rel}, desto mehr weicht die mittlere Auftragsdurchlaufzeit ZDL_m vom Idealwert ab.

$$ZDL_{rel} = \frac{ZDL_m}{ZDL_{min}} \qquad (3)$$

Tabelle 1 Zustandsmerkmale der planungsrelevanten Entitäten.

Planungsrelevante Entität	Zustandsmerkmale
Produktionsauftrag	- Maschinenindividuelle Prozesszeiten der Arbeitsvorgänge - Bearbeitungsstatus der Arbeitsvorgänge
Produktionsmaschine	- Auftragsbestand - Mittelwert und Streuung der Auftragszeit im Bestand - Mittlere Reichweite des Arbeitssystems - Auslastung

Der leistungsorientierte Belohnungsterm R_A verfolgt hingegen das Ziel, Leistungsdifferenzen zu minimieren und die mittlere Auslastung A_m zu maximieren. Hierzu erhält ein Agent nach Formel 4 für Entscheidungen umso größere Belohnungen, je höher die mittlere Auslastung A_m der Maschinen und je niedriger die zugehörige Standardabweichung σ_A ist.

$$R_A = A_m - \lambda_A \sigma_A \quad (4)$$

Der differenzbildende Einfluss der Auslastungsstreuung σ_A wird über den Gewichtsfaktor λ_A gesteuert, der empirisch zu bestimmen ist. Wichtig anzumerken ist, dass die Belohnungen nach Formel 1 nur am Ende eines Simulationsdurchlaufes an einen Agenten ausgegeben werden, da nur nach Zuweisung aller Arbeitsvorgänge die Güte eines Belegungsplans ermittelt werden kann.

Um die Leistungsfähigkeit von Algorithmen systematisch zu bewerten, wird die Simulationsumgebung darüber hinaus als statisch angenommen. In statischen Produktionsumgebungen sind vor Simulationsbeginn alle Produktionsaufträge, Maschinen sowie deren Zuweisungsmöglichkeiten bekannt. Während der Simulationszeit treten keine Veränderungen auf, womit unerwartete Auftragseingänge und Maschinenausfälle unberücksichtigt bleiben. Zudem werden Rüst- und Transportzeiten nicht betrachtet.

Infolge der vereinfachenden Annahmen konzentriert sich die Zustandsraummodellierung auf die Merkmale, die unter statischen Bedingungen die Zuweisungsentscheidung eines Agenten beeinflussen. Der Zustand s eines Agenten setzt sich aus den Merkmalen nach **Tabelle 1** zusammen. In jedem Zustand s muss ein Agent eine Auswahl auf Basis der verfügbaren Handlungsoptionen treffen, die durch den Aktionsraum $A(s)$ repräsentiert werden. Der Aktionsraum $A(s)$ enthält alle kombinatorischen Zuweisungsmöglichkeiten, wobei die Menge an Aktionen zustandsabhängig durch die Ausführbarkeit von Arbeitsvorgängen und die Maschinenverfügbarkeit eingeschränkt wird.

3.3 Deep-Q-Network

Als Deep-Q-Networks [13] werden wertebasierte RL-Verfahren bezeichnet, die ein tiefes neuronales Netz zur Approximation der Q-Funktion $Q(s, a)$ und Lösung eines MEP verwenden.

Charakteristisch für diese Verfahrenskategorie ist die Nutzung von Experience Replay [15]. Hierbei werden die Erfahrungen eines Agenten in einem Zwischenspeicher abgelegt und für das Training eines RL-Modells stichprobenartig abgerufen. Jede Erfahrung wird dabei durch ein Tupel (s, a, r, s') bestimmt, dass den Übergang von einem Zustand s in den Folgezustand s´ beschreibt. Die Aktion a bezeichnet das kausale Ereignis für den Zustandsübergang, das mit einem Belohnungswert r in seiner Güte quantifiziert wird. Das Training von neuronalen Netzen auf Grundlage stochastischer Gradientenverfahren setzt voraus, dass die verwendeten Daten unabhängig und identisch verteilt sind. Experience Replay ermöglicht es, durch das Ziehen von Stichproben aus einem Zwischenspeicher die Korrelation zwischen aufeinanderfolgenden Zustandstransitionen zu relativieren und das Modelltraining zu stabilisieren. Eine weitere Maßnahme zur Stabilisierung des Modelltrainings beschreibt die Verwendung von Target Networks. Unter einem Target Network wird die Kopie eines Deep-Q-Networks verstanden, das in periodischen Intervallen mit den Gewichten des Hauptnetzes aktualisiert wird.

4 Vergleich

Um das Potenzial von RL-Methoden für den Einsatz in der Produktionssteuerung zu bewerten, wurde ein DQN [13] mit Algorithmen aus dem Stand der Technik verglichen. Die Referenzalgorithmen umfassen den CP-SAT Solver [7], die Tabu-Suche [16], den genetischen Algorithmus NSGA-II [17] sowie die Partikelschwarmoptimierung [18]. Zur Untersuchung der Algorithmen wurden ausgewählte Benchmark-Datensätze Kacem 1 und Kacem 2 von *Kacem* et al. [19] verwendet. Als Bewertungskriterien wurden die mittlere Durchlaufzeit ZDL_m, die mittlere Maschinenauslastung A_m sowie der Variationskoeffizient der Auslastung $CV[A]$ herangezogen, die auf Basis des erstellten Maschinenbelegungsplans berechnet wurden. Der Variationskoeffizient $CV[A]$ wird durch das Verhältnis der Auslastungsstreuung σ_A und der mittleren Auslastung A_m bestimmt. Diese Metrik dient als Maß für die Gleichmäßigkeit der Auslastungsverteilung. Je näher der Variationskoeffizient bei null liegt, desto ausgeglichener sind die Arbeitsvorgänge gemessen an den Prozesszeiten auf die verfügbaren Produktionsmaschinen verteilt.

Die Modellarchitektur des DQN setzt sich aus einem Encoder und einem Decoder zusammen. Der Encoder besitzt zwei separate Eingangskanäle, über welche die Auftrags- und Ressourcenmerkmale in eine latente Vektorrepräsentation transformiert werden. Beide Encoder-Komponenten bestehen aus $N_{e,\,l} = 2$ Schichten mit jeweils $N_{e,\,n} = 128$ Neuronen. Die latenten Merkmalsrepräsentationen werden über ein Masked-Mean-Pooling Verfahren zusammengefasst, um einen Vektor fester Größe für den Decoder zu erstellen. Hierbei werden die Mittelwerte über diejenigen Elemente gebildet, die im betrachteten Zustand aktiv sind. Die Aktivität der Auftragsmerkmale wird über die zuweisbaren Arbeitsvorgänge gemäß der technisch bedingten Bearbeitungsreihenfolge bestimmt. Bei den Ressourcenmerkmalen ist hingegen die Verfügbarkeit der Produktionsmaschinen maßgebend. Mithilfe des Decoders wird anschließend die latente Merkmalsrepräsentation zur Bestimmung der Q-Werte genutzt. Der Decoder setzt sich analog zur Encoder-Architektur aus $N_{d,\,l} = 2$ Schichten mit jeweils $N_{d,\,n} = 128$ Neuronen zusammen, die in der Ausgangs-

Tabelle 2 Vergleich von exakten, metaheuristischen und lernbasierten Berechnungsverfahren.

Benchmark			Kacem 1	Kacem 2
Algorithmus	Metrik	Einheit		
CP-SAT	ZDL_m	[-]	12,8	13,6
	A_m	[%]	62,9	95,9
	CV[A]	[-]	0,243	0,054
Tabu-Suche	ZDL_m	[-]	18	41,4
	A_m	[%]	53,6	55,4
	CV[A]	[-]	0,664	0,478
NSGA-II	ZDL_m	[-]	18	21,4
	A_m	[%]	56,8	73,9
	CV[A]	[-]	0,122	0,120
Partikelschwarmoptimierung	ZDL_m	[-]	13,8	24,3
	A_m	[%]	67,9	77,2
	CV[A]	[-]	0,114	0,056
DQN	ZDL_m	[-]	19	35,6
	A_m	[%]	57,3	55,1
	CV[A]	[-]	0,384	0,654

schicht in eine separate Q-Komponente pro Produktionsmaschine überführt werden. Jede Q-Komponente schätzt die Q-Werte aller Aktionen einer Produktionsmaschine. Für das Training des DQN wurden insgesamt Ns = 500 000 Entscheidungssituationen simuliert, im Zwischenspeicher abgelegt und dem Modell in Batch-Größen von N_B = 1 024 zur Anpassung der Gewichte zugeführt. Die Gewichtsanpassung erfolgt mit einer Lernrate, die über den Trainingsverlauf von $\alpha = 0{,}001$ auf $\alpha = 0{,}0002$ linear absinkt. Die Aktualisierung des Target Networks wurde in einem Intervall von N_T = 5 000 simulierten Entscheidungssituationen vorgenommen. Für das Modelltraining wurde das Ray RLlib Framework [20] verwendet. Weitere Hyperparametereinstellungen wurden von der dort verfügbaren DQN-Implementierung übernommen.

In **Tabelle 2** sind die Auswertungen der untersuchten Algorithmen dargestellt. Die Ergebnisse zeigen, dass der CP-SAT Solver im Hinblick auf die Durchlaufzeitminimierung und dem Ausgleich von Leistungsdifferenzen auf einem möglichst hohem Auslastungsniveau den anderen Verfahren überlegen ist. Besonders für die kleine Probleminstanz des Kacem 1 Benchmark-Datensatzes konnte der Solver in nur t_R = 6,7 s die optimale Lösung bestimmen. Für das Kacem 2 Benchmark konnte jedoch ein deutlicher Anstieg der Berechnungszeit festgestellt werden, der auf die erhöhte Planungskomplexität zurückzuführen ist. Während das Kacem 1 Benchmark vier Produktionsaufträge und fünf Maschinen enthält, müssen im Kacem 2 Benchmark bereits zehn Aufträge und sieben Maschinen berücksichtigt werden. Dem gegenüber liefern die untersuchten Metaheuristiken unabhängig von der Planungskomplexität Lösungen in wenigen Sekunden. Die Lösungsgüte der Tabu-Suche, des NSGA-II und der Partikelschwarmoptimierung ist dem CP-SAT Solver jedoch unterlegen. Während die Leistungsunterschiede beim Kacem 1 Benchmark noch moderat ausfallen, werden diese beim Kacem 2 Benchmark infolge der höheren Planungskomplexität signifikant. Metaheuristiken haben folglich den Vorteil, dass sie Lösungen auch für komplexere Problemstellungen konsistent mit kurzen Berechnungszeiten bestimmen können. Dieser Vorteil kommt jedoch zu Kosten der Lösungsgüte. Das DQN weist gegenüber dem CP-SAT ebenfalls deutliche Leistungsdefizite auf. Beim Kacem 1 Benchmark fällt die mittlere Durchlaufzeit ZDL_m des DQN um ΔZDL_m = 48,4 % höher und die mittlere Auslastung A_m um ΔA_m = 5,6 % geringer aus. Der höhere Variationskoeffizient CV[A] des DQN lässt zudem eine ungleichmäßigere Auslastungsverteilung erkennen. Das Leistungsdefizit nimmt bei steigender Planungskomplexität signifikant zu, was anhand der Lösungsgüte des Kacem 2 Benchmarks zu erkennen ist. Die Schwächen des DQN sind insbesondere auf die methodischen Herausforderungen von RL zurückzuführen. Methoden des RL erfordern eine anwendungsfallspezifische Modellierung des Zustands-, Aktionsraums sowie der Belohnungsfunktion, die durch aufwendige Versuche empirisch zu ermitteln sind. Irrelevante Zustandsmerkmale sowie ungünstige Aktionsrepräsentationen können ein RL-Modell daran hindern, die Wirkzusammenhänge zwischen den Merkmalen der Produktionsumgebung und den Auftragszuweisungen korrekt zu erkennen. Darüber hinaus stellt die Auswahl einer geeigneten Modellarchitektur und das stabile Training von tiefen neuronalen Netzen eine zentrale Herausforderung dar. Eine unzureichende Abstimmung der Encoder-, der Decoder-Architektur sowie der Hyperparameter wie Lernrate und Explorationsstrategie kann die Stabilität und die Effizienz des zeitaufwendigen Modelltrainings erheblich beeinträchtigen.

5 Fazit und Ausblick

Die Untersuchungsergebnisse zeigen, dass DQN im betrachteten Zustand nicht konkurrenzfähig zu etablierten Algorithmen aus dem Stand der Technik sind. Während die Leistungsunterschiede zu Metaheuristiken moderat ausfallen, sind deutliche Defizite im Hinblick auf die Durchlaufzeit- und Auslastungsoptimierung zu erkennen. Nichtsdestotrotz liefern weiterführende wissenschaftliche Untersuchungen Indizien für die Leistungsfähigkeit von RL-Methoden [21, 22]. Um das identifizierte Potenzial in die praktische Anwendung zu bringen, bedarf es jedoch weitere Forschung und Entwicklung. Hierzu sind insbesondere erweiterte Modellarchitekturen, wie zum Beispiel Double DQN oder Dueling DQN Ansätze, und Explorationsstrategien zu untersuchen, um die Trainingsstabilität und Konvergenzgeschwindigkeit zu erhöhen. Darüber hinaus gilt es, Methoden zur Modellierung des Zustandsraums und der Belohnungsfunktion experimentell zu evaluieren. Mithilfe von Methoden des State Representation Learning können beispielsweise Merkmale der Produktionsumgebung in latente und semantisch aussagekräftige Zustandsrepräsentationen überführt werden, die eine Approximation optimaler Entscheidungsstrategien unterstützen. Ergänzend dazu können Methoden des Inverse Reinforcement Learning verwendet werden, um aus beobachtetem Expertenverhalten von Produktionsplanern die Belohnungsfunktionen für Agenten zu rekonstruieren. Darüber hinaus rücken Methoden des Multi-Agent Reinforcement Learning zunehmend in den Fokus der Forschung, um die Komplexität des Optimierungsproblems durch eine Dekomposition des Zustands- und Aktionsraums von Agenten zu reduzieren. Von besonderem Interesse ist dabei das explizite Lernen kooperativer Entscheidungsstrategien, das zur Auflösung von Zielkonflikten zwischen autonom agierenden Agenten beitragen kann. Verfahren dieser Art besitzen das Potenzial, eine vollständig dezentrale Steuerung von Produktionsprozessen in der Zukunft zu ermöglichen.

FÖRDERHINWEIS

Dieser Beitrag wird von der Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e. V. durch das interne Fraunhofer-Programm der Leitprojekte gefördert. Die Ergebnisse des Beitrags entstanden im Rahmen des Fraunhofer-Leitprojektes EMOTION.

LITERATUR

[1] Wiendahl, H.-P.: Fertigungsregelung. Logistische Beherrschung von Fertigungsabläufen auf Basis des Trichtermodells. München, Wien: Carl Hanser Verlag, 1997
[2] Xiong, H.; Shi, S.; Ren, D.; Hu, J.: A survery of job shop scheduling problem: The types and models. Computers & Operations Research 142 (2022) 105731
[3] Zhang, X.; Zhu, G.-Y.: A literature review of reinforcement learning methods applied to job-shop scheduling problems. Computers & Operations Research 175 (2025) 106929
[4] Land, A. H.; Doig, A. G.: An automatic method of solving discrete programming problems. Econometrica 28 (1960) 3, S. 497–520
[5] IBM Corp.: IBM ILOG CPLEX Optimization Studio documentation. Firmenschrift. 2024. Internet: https://www.ibm.com/docs/en/icos/22.1.2?announcement=ilog-cplex-optimization-studio-221. Zugriff: 2025-10-02
[6] Gurobi Optimization: Gurobi Optimizer Reference Manual. Firmenschrift. 2024. Internet: https://docs.gurobi.com/projects/optimizer/en/current/index.html. Zugriff: 2025-10-02
[7] Perron, L.; Didier, F.: Cp-sat v9.10. Google. 2024. Internet: https://developers.google.com/optimization/cp/cp_solver/. Zugriff: 2025-10-02
[8] Lödding, H.: Verfahren der Fertigungssteuerung. Berlin, Heidelberg: Springer, 2016
[9] Xie, J.; Gao, L.; Peng, K.; Li, X.; Li, H.: Review on flexible job shop scheduling. IET Collaborative Intelligent Manufacturing 1 (2019) 3, S. 67–77
[10] Chaudry, I. A.; Khan, A. A.: A research survey: review of flexible job shop scheduling techniques. International Transactions in Operational Research 23 (2016) 3, S. 551–591
[11] Türkyilmaz, A.; Senvar, Ö.; Ünal, I.; Bulkan, S.: A research survey: heuristic approaches for solving multi objective flexible job shop problems. Journal of Intelligent Manufacturing 31 (2020), S. 1949–1983
[12] Wang, R.; Wang, G.; Sun, J.; Deng, F.; Chen, J.: Flexible Job Shop Scheduling via Dual Attention Network Based Reinforcement Learning (2024). Internet: https://arxiv.org/abs/2305.05119. Zugriff: 2025-10-02
[13] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M.: Playing Atari with Deep Reinforcement Learning (2013). Internet: https://arxiv.org/abs/1312.5602. Zugriff: 2025-10-02
[14] Sutton, R. S.; Barto, A. G.: Reinforcement Learning. Massachusetts: MIT Press, 2018
[15] Lin, L.-J.: Reinforcement Learning for Robots using Neural Networks. Dissertation. Carnegie Mellon University. 1993
[16] Saidi-Mehrabad, M.; Fattahi, P.: Flexible job shop scheduling with tabu search algorithms. The International Journal of Advanced Manufacturing Technology 32 (2007) 5, S. 563 - 570
[17] Rabiee, M.; Zandieh, M.; Ramezani, P.: Bi-objective partial flexible job shop scheduling problem: NSGA-II, NRGA, MOGA and PAES approaches. International Journal of Production Research 50 (2012) 24, S. 7327–7342
[18] Liu, H.; Abraham, A.; Wang, Z.: A Multi-swarm Approach to Multi-objective Flexible Job-shop Scheduling Problems. Fundamenta Informaticae 95 (2009) 4, S. 465–489
[19] Kacem, I.; Hammadi, S.; Borne, P.: Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. Mathematics and Computers in Simulation 60 (2002) 3 – 5, S. 245–276
[20] Liang, E.; Liaw, R.; Moritz, P.; Nishihara, R.; Fox, R.; Goldberg, K.; Gonzalez, J.; Jordan, M.; Stoica, I.: RLlib: Abstractions for Distributed Reinforcement Learning (2018). Internet: https:// https://arxiv.org/pdf/1712.09381. Zugriff: 2025-10-02
[21] Liu, C.; Chen, K.; Wang, H.; Yang, B.; Leng, J.: Job shop scheduling by Deep Dual-Q Network with Prioritized Experience Replay for resilient production control in flexible manufacturing system. Computers & Operations Research 183 (2025) 107190
[22] Belmamoune, M. A.; Ghomri, L.; Yahouni, Z.: Solving a Job Shop Scheduling Problem Using Q-Learning Algorithm. In: SOHOMA 2022 12th international workshop on service oriented, holonic and multi-agent manufacturing systems for industry of the future, Bukarest, Rumänien, September 2022, S. 196–209

Prof. Dr. h. c. Dr.-Ing. Eckart Uhlmann

Prof. Dr.-Ing. Julian Polte

Dipl.-Ing. Christopher Mühlich
christopher.muehlich@ipk.fraunhofer.de

Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik IPK
Pascalstr. 8–9, 10587 Berlin
www.ipk.fraunhofer.de