DOI: 10.2195/lj_proc_noortwyck_de_202211_01
URN: urn:nbn:de:0009-14-55977

Entwicklung eines DRL-Agenten zur Reihenfolgeoptimierung für Hochregallager mit Shuttle-Fahrzeugen

Development of a DRL agent for sequence optimisation for high-rack warehouses with shuttle vehicles

Ruben Noortwyck
Robert Schulz

Institut für Fördertechnik und Logistik
Fakultät für Konstruktions-, Produktions- und Fahrzeugtechnik
Universität Stuttgart

Aufgrund steigender Dynamik und Heterogenität in der Produktion sind auch die Anforderungen an die Intralogistik und im speziellen an Lagersysteme gestiegen. Lagersysteme müssen flexibel sein und einen hohen Durchsatz ermöglichen. Diese Anforderungen werden durch Shuttlesysteme erfüllt. Damit der Durchsatz von Shuttlesystemen softwarebasiert weiter gesteigert werden kann, wurden Konzepte entwickelt, welche mit Deep Reinforcement Learning (DRL) die Blockaden, welche z. B. beim Gassenwechsel oder bei mehreren Auslagerungen in einer Gasse auftreten, durch eine genänderte Auslagerungsreihenfolge minimieren. Die bisher entwickelten Konzepte betrachten ausschließlich eine sehr kleine Anzahl an Lagerplätzen. Reale Shuttlesysteme verfügen teilweise über mehrere tausend Lagerplätze pro Ebene. Daher wird in diesem Beitrag ein DRL-Konzept entwickelt, welches in einem realen Shuttlesystem die Auslagerungsreihenfolge anpasst, um durch eine Minimierung der Blockaden eine Durchsatzsteigerung zu erreichen.

[Schlüsselwörter: Deep Reinforcement Learning, Künstliche Intelligenz, Shuttle-Systeme, Durchsatzoptimierung]

Due to increasing dynamics and heterogeneity in production, the demands on intralogistics and especially on storage systems have increased. Storage systems must be flexible and enable a high throughput. These requirements are fulfilled by shuttle systems. To be able to increase the throughput of shuttle systems on a software basis, concepts have been developed that use Deep Reinforcement Learning (DRL) to minimise the blockages that arise, e.g. when changing gears or when several withdrawals are made in one gear, by changing the retrieval sequence. These concepts only consider a very small number of storage locations. Real shuttle systems sometimes have several thousand storage locations per level. Therefore, this paper develops a DRL concept that adapts the retrieval sequence in a real shuttle system to minimise blockades and increase throughput.

[Keywords: Deep Reinforcement Learning, Artificial Intelligence, AVS/RS, throughput optimization]

1 EINLEITUNG

Shuttlesysteme werden immer häufiger in der Praxis eingesetzt, da diese den hohen Anforderungen an Flexibilität, Durchsatz und Energieeffizienz gerecht werden.

Ein Shuttlesystem ist ein automatisiertes Lager, in dem meist Kleinladungsträger mit autonomen Fahrzeugen (Shuttle-Fahrzeuge) und Aufzügen transportiert werden. Dadurch wird, im Gegensatz zu Lagern mit krangestützten Regalbediengeräten, der vertikale und horizontale Transportprozesse getrennt. [1]

Je nach Systemauslegung können sich die Fahrzeuge innerhalb des Shuttlelagers frei bewegen und nutzen dadurch die gleichen Fahrwege und Lifte. Aufgrund der durch die Blockaden resultierenden Wartezeit findet eine Verlängerung der Spielzeit statt, welche wiederrum eine Durchsatzreduzierung zur Folge hat. So wurde z. B. in [2] nachgewiesen, dass sich durch das gegenseitige Blockieren der Shuttle-Fahrzeuge die Spielzeit um bis zu 20 % erhöht.

Die Anzahl der auftretenden Blockaden ist von der Auslagerungsreihenfolge abhängig, da diese vorgibt, wann welches Fahrzeug einen bestimmten Weg befahren muss. Daher lässt sich, wie bereits in [3] nachgewiesen, durch eine veränderte Auslagerungsreihenfolge der Durchsatz erhöhen.

In [3] wurde das Deep Reinforcement Learning (DRL) genutzt, um die Auslagerungsreihenfolge zu bestimmen. Das DRL ist eine Verbindung des Reinforcement Learning (RL) und des Deep Learning (DL). Das RL kann durch das

DOI: 10.2195/lj_proc_noortwyck_de_202211_01
URN: urn:nbn:de:0009-14-55977

Tätigen von Aktionen in einer Lernumgebung einen Rückgabewert maximieren und dadurch eine Strategie erlernen [4]. Das DL ermöglicht es Computersystemen aus Daten und Erfahrungen, durch das Verwenden von tiefen neuronalen Netzen, zu lernen [5].

Das in [3] betrachtete Shuttlelager verfügt ausschließlich über 100 Lagerplätze auf einer Ebene. Die gentrennte Betrachtung einer Ebene ist für das Auflösen von Blockaden ausreichend. Jedoch verfügen Shuttlesysteme in der Realität über mehrere tausend Lagerplätze pro Ebene.

Daher wird in diesem Beitrag ein Konzept eines DRL-Agenten vorgestellt, welches in einem realen Shuttlesystem, durch eine geänderte Auslagerungsreihenfolge, den Durchsatz erhöhen soll. Neben der höheren Anzahl an Lagerplätzen pro Ebene werden sowohl Einlagerungen als auch Umlagerungen betrachtet. Zur Überprüfung des Konzepts wurde eine Ebene eines realen Shuttlesystems inkl. Steuerung in der Simulationssoftware Plant Simulation modelliert. Der DRL-Agent wurde mit Keras-RL und TensorFlow umgesetzt.

In den folgenden Kapiteln werden zuerst Grundlagen der Shuttlesysteme sowie des DRL erläutert. Anschließend wird das betrachtete reale Shuttlesystem inkl. Besonderheiten beschrieben. Es folgt die Beschreibung des Simulationsmodells sowie die Ermittlung des durchschnittlichen Optimierungspotenzials für unterschiedliche Lagerkonfigurationen. Anschließend wird die DRL-Architektur vorgestellt sowie bisherige Erkenntnisse und Ergebnisse präsentiert.

2 SHUTTLESYSTEME

Shuttlesysteme lassen sich in Abhängigkeit ihres Bewegungsraums in gassengebunden, ebenengebunden, gassen- und ebenengebunden sowie freie Systeme einteilen.

Innerhalb gassengebundener Systeme ist es den Shuttle-Fahrzeugen möglich, über einen Lift die Ebene zu wechseln. Ein Quergang, zum Wechseln von Ebenen ist bei dieser Ausprägung nicht vorhanden. [6]

Bei ebenengebundenen Shuttlesystemen ist es den Shuttle-Fahrzeugen möglich über Quergänge die Gasse zu wechseln. Ein wechseln der Ebene ist nicht möglich. Ebenengebundene Shuttlesysteme verfügen über einen Behälterlift. [6]

Eine Kombination der bereits genannten Ausprägungen bilden gassen- und ebenengebundene Shuttlesysteme. In diesen Systemen ist es den Shuttle-Fahrzeugen nicht möglich die Gasse über Quergänge oder die Ebene über einen Shuttlelift zu wechseln. [6]

Die größte Flexibilität bieten ungebundenen Shuttlesysteme. Diese Ausprägung ermöglicht den Shuttle-Fahrzeugen sowohl einen Gassenwechsel als auch einen Ebenenwechsel. Durch das Einschleusen weiterer Shuttle-Fahrzeuge kann der Durchsatz dieser Systeme bis zu einem gewissen Grad erhöht werden. [6]

Shuttlesysteme sind meist über Fördertechnik mit Kommissionierarbeitsplätzen verbunden. Dabei ist es wichtig, dass die in unterschiedlichen Gassen und Ebenen lagernden Behälter in der richtigen Sequenz am Kommissionierarbeitsplatz ankommen. Hierfür wird meist eine komplexe und teure Fördertechnik mit u. a. Sequenziertürmen benötigt.

Neuartige Konzepte, wie z. B. im GEBHARDT StoreBiter® 300 One Level Shuttle X (OLS X) oder im ADAPTO von Vanderlande verfügen bereits im Shuttlesystem über die Möglichkeit der Sortierung und Sequenzierung von Behältern.

3 DEEP REINFORCEMENT LEARNING

Das Deep Reinforcement Learning (DRL) lässt sich dem Maschinellem Lernen zuordnen und kombiniert das Reinforcement Learning (RL) und das Deep Learning (DL). Dabei trifft, wie im RL üblich, ein Agent die Entscheidungen und lernt durch „Versuch und Irrtum“ die optimale Strategie für das vorliegende Problem.

Beim RL interagiert der Algorithmus, welcher als Agent bezeichnet wird, zum Lernen mit der Umgebung. Das RL besteht aus fünf Hauptelementen, welche im Folgenden beschrieben werden. [4]

- Action (Aktion):

  Der Agent wählt aus den vorgegebenen Handlungsalternativen eine aus.

- Status (Zustand):

  Situation der Umgebung, welche der Agent zu einem bestimmten Moment übermittelt bekommt.

- Environment (Umgebung):

  Gibt die Rahmenbedingungen für den Agenten vor. Abhängig von einer gewählten Aktion wird der aktuelle Zustand der Umgebung in den nächsten Zustand überführt.

- Reward (Rückgabewert):

  Ein positiver oder negative Wert, den der Agent, basierend auf der gewählten Aktion, bekommt. Der Rückgabewert bewertet die getätigte Aktion und hilft dem Agenten die Qualität der getätigten Aktion einzuschätzen.

DOI: 10.2195/lj_proc_noortwyck_de_202211_01
URN: urn:nbn:de:0009-14-55977

- Strategy (Strategie):

  Abhängig von der Strategie wird die nächste Aktion mit der höchsten zu erwartenden Belohnung zu einem bestimmten Zustand gewählt.

Es existieren verschiedene Lernalgorithmen für das RL. Ein wichtiger Lernalgorithmus ist das Q-Learning. Dabei werden Q-Werte verwendet, um die Strategie und somit das Verhalten des Lernalgorithmus schrittweise zu verbessern. Die Q-Werte, welche die Qualität einer Aktion A in einem Zustand S (Q(S, A)) darstellen, werden in einer Q-Tabelle gespeichert und wieder ausgelesen.

Die Q-Werte werden durch den Rückgabewert, welcher der Agent durch die Interaktion mit der Umgebung erhält, sukzessive aktualisiert. [4]

Bei besonders komplexen Problemen, welche durch herkömmliche RL-Algorithmen, wie z. B. das Q-Learning, nicht gelöst werden können, wird das DRL angewendet. Dabei wird, wie im DL, ein tiefes neuronales Netz verwendet, welches für jede mögliche Aktion Input- und Output-Werte enthält und so die passende Strategie ermittelt. [5]

Neuronale Netze bestehen aus mehreren miteinander verbundenen Knoten, auch Neuronen genannt. Die Architektur eines neuronalen Netzes ist durch die Anzahl an verdeckten Schichten, die Anzahl an Neuronen pro Schicht sowie die Verbindungsart der einzelnen Neuronen definiert. Ein neuronales Netz, welches über mindestens zwei verdeckte Schichten verfügt, wird als tiefes neuronales Netz bzw. Deep Neuronal Network (DNN) bezeichnet. [5]

Neuronale Netze lassen sich unterteilen in Feedforward-Netze und Rekurrente-Netze. Bei Feedforward-Netzen fließen die Informationen von der Input-Schicht zur Output-Schicht. Es gibt keine Rückkopplung zwischen den einzelnen Neuronen. Diese neuronalen Netze sind im Vergleich zu anderen Netzen wenig komplex. Ein weit verbreitetes Feedforward-Netz, welches im DL eingesetzt werden kann, ist dass Deep Feed Forward Netz(DFF). Über eine Rückkopplung zwischen einzelnen Neuronen verfügen die rekurrenten neuronalen Netze. Dabei kann ein Neuron an sich selbst, an Neuron derselben Schicht oder an Neuron in einer anderen Schicht rückkoppeln. Im Gegensatz zu Feedforward-Netzen sind rekurrente neuronale Netze nicht auf einen zusammengehörigen Eingabe- und Ausgabewert beschränkt. Sie können umfassendere sowie komplexere Probleme mit zusammenhängenden Sequenzen von Eingaben und Ausgaben erlernen. [5]

Neuronale Netze lernen durch das Minimieren einer Fehlerfunktion, welche die Differenz zwischen den geschätzten und tatsächlichen Ergebnissen des Netzes angibt. Das Netzwerk kann durch Backpropagation gelernt werden, indem die Gewichtung und die Basiswerte angepasst werden. Anfänglich werden die Gewichte mit Zufallswerten zwischen 0 und 1 oder -1 und 1 initialisiert. Der Backpropagation-Algorithmus besteht aus dem Vorwärtsschritt (forward pass) und dem Rückwärtsschritt (backward pass), welche solange wiederholt werden bis das Ziel erreicht wurde oder keine Verbesserung mehr erreicht wird. [7]

Beim Deep-Q-Learning ersetzt ein neuronales Netz die Q-Tabelle des bereits beschriebenen Q-Learning. Die Q-Werte werden in einer Wertefunktion, welche abhängig von der Aktion a, einem Zustands s und der Gewichtung θ des neuronalen Netzes ist, als $Q(s, a; \theta)$ dargestellt. Die Q-Werte für jede mögliche Aktion werden als Output, in Abhängigkeit vom Zustand, welcher als Input verwendet wird, durch das neuronale Netz ausgegebene. [8]

4 Referenzsystem und Simulation

Das Referenzsystem, welches als Simulationsmodell abgebildet wurde und zum Training des neuronalen Netzes genutzt wird, ist das von GEBHARDT Fördertechnik entwickelte StoreBiter® OLS X. Das StoreBiter OLS X ist ein hochflexibles und skalierbares Shuttle-System, welches zielgerichtet an unterschiedlichste, sich ändernde Marktsituationen, Prozesse und Unternehmensanforderungen angepasst werden kann.

Im Folgenden werden die Besonderheiten des StoreBiter® OLS X, auftretende Blockaden beim Gassenwechsel sowie der Aufbau des Simulationsmodells beschrieben.

4.1 Besonderheiten des Referenzsystems

Das betrachtete Referenzsystem besteht aus zwei oder mehreren Gassen sowie aus mehreren Ebenen. Abhängig von der benötigten Leistung befinden sich ein oder mehrere Shuttle-Fahrzeuge auf einer Ebene. Die Besonderheit des StoreBiter® OLS X Shuttlesystems sind die auf jeder Ebene am Gasseneingang vorhandenen Plattformen, welche in Abbildung 1 dargestellt sind. Diese Plattformen ermöglichen einen freien, schienenlosen und unkomplizierten Gassenwechsel der Shuttle-Fahrzeuge. In den einzelnen Gassen fahren die Shuttle-Fahrzeuge wiederum auf Schienen. Auf den Plattformen des StoreBiter® OLS X werden die Shuttle-Fahrzeuge durch eine integrierte optische Navigation gesteuert und erreicht dadurch zielsicher die vorgegebene Position.

Einer der Vorteile dieses Shuttlesystems ist, dass jeder Kommissionierarbeitsplatz direkten Zugriff auf jeden Artikel im kompletten Lager und in der exakt benötigten Sequenz hat. Jedes Shuttle-Fahrzeug kann die Gasse verlassen und die Heber eines jeden Arbeitsplatzes direkt anfahren, wodurch eine Versorgung mit Behälter ohne Zeitverlust garantiert werden kann.

DOI: 10.2195/lj_proc_noortwyck_de_202211_01
URN: urn:nbn:de:0009-14-55977

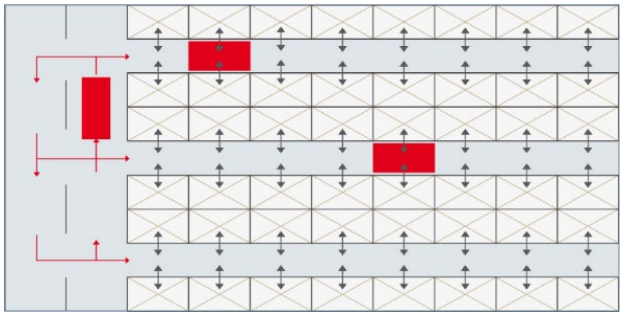

Abbildung 1. Schienenloser Gassenwechsel durch vorgelagerte Plattform. © GEBHARDT Fördertechnik GmbH

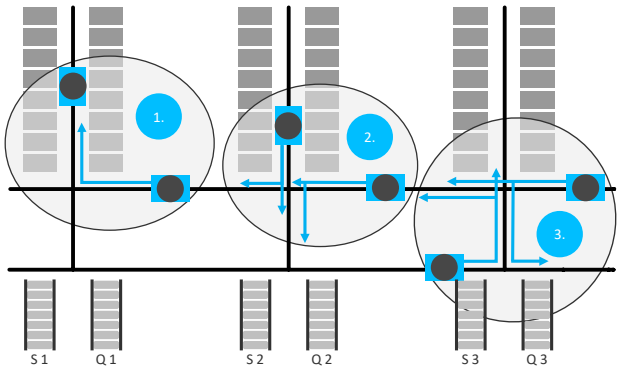

Abbildung 2. Mögliche Fahrsituationen und potenzielle Blockaden der Shuttle-Fahrzeuge

4.2 Blockaden

Der in Abschnitt 4.1 beschriebene Gassenwechsel auf den vorhandenen Plattformen führt, abhängig von der Fahrsituation, zu Blockadesituationen. Damit die Shuttle-Fahrzeuge auf der Plattform nicht kollidieren, reservieren diese sich die zwei darauffolgenden Wegabschnitte. Ein Überholen von Fahrzeugen ist auf der Plattform nicht möglich. Dadurch ergeben sich in den folgenden drei Fahrsituationen Wartezeiten.

(1) Mehrere Auslagerungen pro Gasse

Bei der Auslagerung von zwei oder mehreren Ladungsträgern aus derselben Gasse, müssen die weitere Fahrzeuge vor der Gasse warten. Dadurch entsteht eine Blockade auf der Plattform.

(2) Fahrwegüberschneidungen im Gassenbereich

Sollte ein Fahrzeug die Gasse verlassen wollen und gleichzeitig ein anderes Fahrzeug dessen Weg queren, muss das Fahrzeug, welches die Gasse verlassen möchte, warten, damit auf der Plattform keine Blockade entsteht.

(3) Fahrwegüberschneidungen auf der Plattform

Auf der Plattform queren sich die Wege der Shuttle-Fahrzeuge in den Kreuzungsbereichen immer wieder, weshalb Fahrzeuge warten müssen. Dies führt zu Blockaden und Wartezeiten bei den einzelnen Fahrzeugen.

Die drei beschriebenen Fahrsituationen, in denen es zu Wartezeiten der einzelnen Shuttle-Fahrzeuge kommen kann, sind in Abbildung 2 dargestellt.

Durch eine veränderte Auslagerreihenfolge können die Blockadesituationen minimiert werden. Abhängig von der vorhandenen Lagerkonfigurationen treten die genannten Fahrsituationen unterschiedlich oft auf.

4.3 Simulationsmodell

Das zum Trainieren des neuronalen Netzes verwendete Simulationsmodell des StoreBiter® OLS X ist flexibel aufgebaut, sodass unterschiedliche Lagerkonfigurationen mit variierender Anzahl an Kommissionierarbeitsplätzen simuliert werden können. Das Grundmodell orientiert sich an einem realen, in der Praxis aufgebauten Shuttlelager mit insgesamt fünf Gassen sowie sechs Arbeitsplätzen und verfügt bei einer doppeltiefen Einlagerung über 560 Lagerplätze pro Gasse. Die Position der Arbeitsplätze an der Plattform ist in Abbildung 2 ersichtlich. Die Anzahl der Shuttle-Fahrzeuge pro Ebene lässt sich ebenfalls flexibel gestalten.

Damit die in Abschnitt 4.2 beschriebenen Fahrsituationen nicht zu einem dauerhaften blockieren führen, wurden die folgenden Regeln in der Simulation festgelegt.

(1) Ist die Lagergasse belegt, muss ein Fahrzeug mit der gleichen Zielgasse auf der Plattform warten, bis die Zielgasse frei ist. (Fahrsituation (1))

(2) Shuttle-Fahrzeuge auf der Plattform haben, gegenüber den Fahrzeugen in der Gasse, Vorfahrt. (Fahrsituation (2)) Voraussetzung ist, dass nicht Fahrsituation (1) vorliegt.

(3) An Kreuzungen auf der Plattform hat das Fahrzeug, welches länger auf die Weiterfahrt wartet Vorfahrt. (Fahrsituation (3))

Das Auflösen der Blockaden benötigt, abhängig von der Fahrsituationen, unterschiedlich viel Zeit. Damit eine Durchsatzerhöhung erreicht wird, müssen die Blockaden reduziert werden. Optimal wäre das vollständige verhindern der Blockaden, welches durch den Einsatz eines einzelnen Shuttle-Fahrzeugs erreicht wird, aber meist unrealistisch ist, da der Durchsatz mit einem einzelnen Shuttle-Fahrzeug zu gering ist.

DOI: 10.2195/lj_proc_noortwyck_de_202211_01
URN: urn:nbn:de:0009-14-55977

5 OPTIMIERUNGSPOTENZIAL

Anhand fünf unterschiedlicher Lagerkonfigurationen, welche in Tabelle 1 dargestellt sind, wurde der Einfluss von Blockaden sowie der Auslagerreihenfolge auf die Gesamtbearbeitungszeit von 600 Auslagerungen in einer zufälligen Reihenfolge untersucht. Dabei wurden für jede Lagerkonfiguration mit einem (es treten keine Blockaden auf) und mit drei Shuttle-Fahrzeugen 1.000 Simulationsläufe durchgeführt. Für alle Konfigurationen wurden sechs Arbeitsplätze verwendet sowie ein Lagerfüllgrad von 95 % festgelegt. Die Durchlaufzeit wird dabei pro Doppelspiel betrachtet. Abhängig von der Lagerkonfiguration ergeben sich unterschiedliche Optimierungspotenziale.

In Konfiguration eins, der ursprünglichen Ebene des betrachteten realen Shuttlelagers, ergibt sich ein durchschnittliches Optimierungspotenzial von 10,15 %. Das maximale Optimierungspotenzial liegt bei 18,10 %. In Abhängigkeit der Gassenanzahl sowie der Gassenlänge ergeben sich unterschiedliche Optimierungspotenziale. Die Simulation zeigt auf, dass bei einer gleichbleibenden Anzahl an Lagerplätzen das Optimierungspotenzial steigt, desto kleiner die Anzahl an Gassen ist. Außerdem steigt das Optimierungspotenzial in Abhängigkeit der Gassenlänge. Dies ist damit begründet, dass bei einer Anzahl von z. B. drei Shuttle-Fahrzeugen und zwei Gassen meist ein Shuttle-Fahrzeug vor einer Gasse warten muss. Zudem steigt die Fahrzeit innerhalb der Gasse und somit auch die Wartezeit am Gasseneingang in Abhängigkeit der Gassenlänge. In Tabelle 2 sind die einzelnen Optimierungspotenziale sowie unterschiedliche Doppelspielzeiten dargestellt.

Tabelle 1. *Betrachtete Lagerkonfigurationen zur Ermittlung des Optimierungspotenzials*

Konfiguration	Gassen	Gassentiefe	Lagerplätze
K1	5	140	2.800
K2	5	56	1.120
K3	4	70	1.120
K4	3	93	1.116
K5	2	140	1.120

Tabelle 2. *Ergebnisse und Optimierungspotenzial der unterschiedlichen Lagerkonfigurationen*

Konfiguration	K1		K2		K3		K4		K5	
Anzahl Shuttle	1	3	1	3	1	3	1	3	1	3
Ø Doppelspielzeit [m:ss]	2:37	2:54	1:54	2:14	1:58	2:25	2:06	2:46	2:29	3:45
Min. Doppelspielzeit [m:ss]	2:31	2:45	1:52	2:09	1:56	2:19	2:04	2:39	2:26	3:34
Max. Doppelspielzeit [m:ss]	2:42	3:05	1:56	2:21	2:00	2:30	2:08	2:55	2:31	3:56
Ø Optimierungspotenzial	10,15 %		15,20 %		18,59 %		24,36 %		32,06 %	
Max. Optimierungspotenzial	18,10 %		21,07 %		23,09 %		29,22 %		38,13 %	

6 DRL-ARCHITEKTUR

Die DRL-Architektur besteht aus dem bereits beschriebenem Simulationsmodell und dem DQN-Agenten. Als neuronales Netz wird aktuell ein DFF mit zwei verborgenen Schichten verwendet. Jede verborgene Schicht besteht aus 512 Neuronen. Die beschriebene DRL-Architektur ist in Abbildung 3 inkl. der Schnittstelle zur Simulationssoftware dargestellt.

Die in Abschnitt 3 beschriebenen Elemente des DRL werden zur Optimierung der Durchlaufzeit für das betrachtete Referenzsystem folgendermaßen definiert.

Action:

Der Agent wählt für das anfragende Shuttle-Fahrzeug, in Abhängigkeit des höchsten zu erwartenden Nutzens (Q-Wert), einen noch abzuarbeitenden Auftrag aus. Die

DOI: 10.2195/lj_proc_noortwyck_de_202211_01
URN: urn:nbn:de:0009-14-55977

Auftragsnummer wird anschließend an das Simulationsmodell übermittelt und der Auftrag durchgeführt.

Status:

Das Simulationsmodell übermittelt den Zustand des Beobachtungsraums an den Agenten. Der Beobachtungsraum wird in Form einer Zustandsmatrix zu dem Zeitpunkt übergeben, wenn ein Shuttle-Fahrzeug einen neuen Auftrag anfragt. Die Zustandsmatrix muss alle relevanten Informationen erhalten, um den nächsten Auftrag auswählen zu können. Dabei gibt es zwei Möglichkeiten, wie die Zustandsmatrix aufgebaut ist.

(a) Die Zustandsmatrix erhält für die nächsten sechs Auslagerungen pro Arbeitsplatz die folgenden Informationen: (1) Verfügbarkeit des Auftrags, (2) Position des anfragenden Fahrzeugs, (3) Position aller anderen Shuttle-Fahrzeuge, (4) Ziele der anderen Shuttle-Fahrzeuge, (5) Auftragsnummer, (6) Zielarbeitsplatz des Auftrags, (7) Lagerplatz des Auftrags.

(b) Die Zustandsmatrix erhält für jeden Lagerplatz einer Ebene für die Auslagerung relevante Informationen. Die Statusmatrix enthält die Informationen (2) – (7) der Variante (a). Folgende Informationen werden im Vergleich zu Variante (a) ergänz oder verändert: (1) Auftrag für Lagerplatz vorhanden und (8) Lagerplatz frei bzw. belegt.

Die genannten Informationen werden für jeden Auslagerauftrag bzw. jeden Lagerplatz in einer Zeile der Matrix dargestellt. Sollte sich aufgrund schwankender Auslagerungen pro Auftrag die Dimension der Zustandsmatrix in Variante (a) ändern, werden Null-Zeilen eingefügt, damit bei jeder Zustandsübermittlung die Dimension der Matrix konstant ist. Die Dimension der Zustandsmatrix der Variante (b) ist von der Anzahl der Lagerplätze abhängig und somit gleichbleibend. Damit die Werte beim Training des neuronalen Netzes nicht als Bewertung wahrgenommen werden, müssen diese z. B. auf 100 normiert werden.

Environment:

Als Trainingsumgebung (Enviroment) wird das in Abschnitt 4.3 beschriebene Simulationsmodell einer Ebene des Referenzsystems verwendet.

Reward:

Für die Auswahl des Auftrags wurden Restriktionen festgelegt, die der Agent erlernen muss. Abhängig von diesen Restriktionen und des gewählten Auslagerauftrags erhält der Agent eine Belohnung (Reward). Innerhalb eines gesamten Auftrags muss eine weiche Sequenz der Auslagerreihenfolge eingehalten werden, ist dies nicht der Fall, fällt die Belohnung stark negativ aus. Weiter muss jeder Auslagerauftrag an den richtigen Arbeitsplatz geliefert werden. Wird der Auslagerauftrag an einen falschen Arbeitsplatz geliefert wird der Agent mit einem negativen Reward bestraft. Da jeder Auslagerauftrag einmalig ausgeführt werden darf, erhält der Agent bei doppelter Ausführung ebenfalls einen stark negativen Rückgabewert. Werden alle Restriktionen eingehalten, wird ein Rückgabewert in Abhängigkeit der Durchlaufzeit ermittelt.

Strategy:

Die Strategie legt die optimale Auslagerreihenfolge fest und wird mit Hilfe des Agenten und des neuronalen Netzes erlernt.

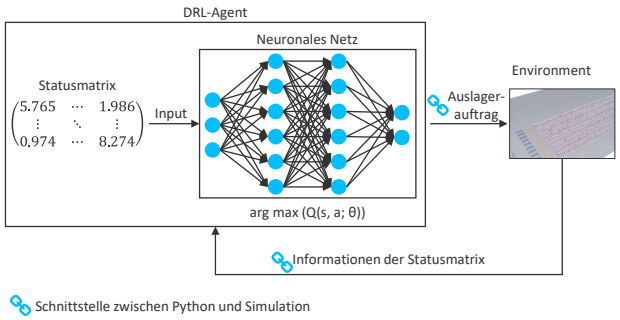

Abbildung 3. DRL-Architektur und Verbindungen zum Simulationsmodell

7 Umsetzung und Bisherige Erkenntnisse

Die im vorherigen Abschnitt aufgezeigte DRL-Architektur wurde mit Keras-RL sowie TensorFlow in Python umgesetzt. Das Simulationsmodell wurde mit der Software Plant Simulation erstellt. Eine Kommunikation zwischen Plant Simulation und dem Python-Skript wurde über eine COM-Schnittstelle realisiert.

Die beiden in Abschnitt 6 beschriebenen Möglichkeiten der Zustandsmatrix wurden umgesetzt und mehrere Trainingsläufe, mit 5.000, 10.000, 15.000 und 20.000 Episoden, durchgeführt. Innerhalb einer Episode mussten 100 Auslagerungen bzw. max. 500 Steps durchgeführt werden. Es zeigte sich, dass die Anzahl an Steps, die benötigt wurden, um die geforderten 100 Auslagerungen durchzuführen, sich kontinuierlich verringerte. Daraus lässt sich schlussfolgern, dass der DRL-Agent lernt, welche Auslagerungen noch verfügbar sind und welche bereits abgearbeitet wurden. Weiter erwies sich die Variante (b) aufgrund sehr langer Trainingszeiten als nicht praktikabel. Die Ursache dafür ist die mit 2.800 Zeilen sehr große Dimension der Zustandsmatrix. Dies stellt für den eigentlichen Trainingsprozess des DRL-Agenten kein Problem dar. Jedoch muss die Zustandsmatrix nach jeder Aktion durch die Simulation vollständig neu übertragen werden. Dafür ist die Geschwindigkeit der COM-Schnittstelle von Plant Simulation nicht ausreichend und verzögerte das Training des Agenten stsrk. Daher muss die Zustandsmatrix so klein wie möglich

DOI: 10.2195/lj_proc_noortwyck_de_202211_01
URN: urn:nbn:de:0009-14-55977

definiert werden. Weitere Schnittstellen, wie z. B. die C-Schnittstelle, wurden bisher nicht verwendet.

Die bisher durchgeführten Experimente mit der Zustandsmatrix (a) zeigen eine Durchsatzoptimierung von 2 – 3 % und liegen damit unter dem in Abschnitt 5 ermitteltem durchschnittlichen Optimierungspotenzial von über 10 % bei der kompletten Ebene (K1) des Referenzsystems. Auch das in [3] durch einen DRL-Agenten erzielte durchschnittliche Optimierungspotenzial von 7 – 8 % konnte für das betrachtete Shuttlelager bisher nicht erreicht werden.

Eine Ursache für die geringe Durchsatzoptimierung könnte das Fehlen wichtiger Informationen für den DRL-Agenten sein. Aufgrund der doppeltiefen Einlagerung ist bei gassenfernen Lagerplätzen eine Umlagerung notwendig. Bisher wird in der Zustandsmatrix dem DRL-Agenten die Position des Auslagerauftrags inkl. gassennah oder gassenfern übergeben. Somit kann der Agent lernen, dass bei gassenfernen Lagerplätzen eine Umlagerung durchgeführt werden muss. Genaue Informationen darüber, wie lange die Umlagerung dauern wird, sind jedoch nicht in der Zustandsmatrix enthalten. Durch eine Erweiterung der Zustandsmatrix soll die Durchsatzoptimierung des DRL-Agenten an die in Abschnitt 5 ermittelten durchschnittlichen Optimierungspotenziale angenähert werden.

8 Zusammenfassung und Ausblick

Im Rahmen dieses Beitrags wurde ein Konzept zur Durchsatzoptimierung von Shuttlesystemen mit einem DRL-Agenten vorgestellt. Dabei sollen in einem Shuttlesystem vorkommende Blockaden durch eine Änderung der Auftragsreihenfolge vermieden werden. Bisher vorhandene Ansätze zeigen, dass eine Durchsatzoptimierung bei einer geringen Anzahl von Lagerplätzen möglich ist. In diesem Beitrag wurde ein reales Shuttlesystem in einem Simulationsmodell abgebildet und das mögliche Optimierungspotenzial ermittelt. Weiter wurden zwei unterschiedliche Möglichkeiten zum Aufbau der Zustandsmatrix erläutert. Beide Möglichkeiten wurden umgesetzt und zum Trainieren eines neuronalen Netzes genutzt. Dabei konnte eine Durchsatzoptimierung von 2 – 3 % erreicht werden. Diese entspricht jedoch nicht dem vorhandenem Optimierungspotenzial. Die Trainingsläufe haben aufgezeigt, dass der Agent schnell lernt, welche Auslagerungen bereits durchgeführt wurden und welche noch verfügbar sind.

Im weiteren Forschungsverlauf muss untersucht werden, ob durch eine Erweiterung der Zustandsmatrix um wichtige Informationen die Durchsatzoptimierung weiter gesteigert werden kann. Weiter müssen Parameter wie die Lernrate angepasst werden. Darüber hinaus können Änderungen an dem neuronalen Netz, wie z. B. die Anzahl an Neuronen pro verdeckter Schicht, durchgeführt werden. Darüber hinaus könnte ein anderes neuronales Netz, wie das bisher verwendet DFF, implementiert werden.

Es muss untersucht werden, ob die gewählte Art der Belohnung, welche abhängig von der absoluten Doppelspielzeit inkl. möglicher Umlagerungen ist, für das Lernen des DRL-Agenten geeignet ist. Eine andere Möglichkeit wäre das Bilden von Vergleichszeiten zur Bewertung der gewählten Aktion.

Gefördert durch:

aufgrund eines Beschlusses
des Deutschen Bundestages

DOI: 10.2195/lj_proc_noortwyck_de_202211_01
URN: urn:nbn:de:0009-14-55977

LITERATUR

[1] D. Arnold und K. Furmans, Materialfuss in Logistiksystemen, Springer Vieweg, Berlin, Heidelberg, 2019.

[2] D. Roy, A. Krishnamurthy, S. Heragu, und C. Malmborg, „Vehicle Interference Effects in Warehousing Systems with Autonomous Vehicles" 6th annual IEEE International Conference on Automation Science and Engineering, Toronto, Kanada, S. 674–679, 2010.

[3] F. Schloz, T. Kriehn, R. Schulz, und M. Fittinghoff, „Entwicklung einer KI-basierten Reihenfolgestrategie für Hochregallager mit autonomen Fahrzeugen" Logistics Journal: Proceedings, 2019.

[4] R. S. Sutton und A. G. Barto, Reinforcement learning: An introduction, MIT press, Camebridge, London, 2018.

[5] I. Goodfellow, B. Yoshua und C. Aaron, Deep learning. MIT press, Camebridge, London, 2016.

[6] K.-H. Wehking, Technisches Handbuch Logistik 1 – Fördertechnik, Materialfluss, Intralogistik, Springer Vieweg, Berlin, 2020.

[7] R. Kruse, S. Mostaghim, C. Borgelt, C. Braune, M. Steinbrecher, Computational Intelligence: A Methodological Introduction, Springer International Publishing, 2022.

[8] R. Atienza, Advanced Deep Learning with Keras: Apply deep learning techniques, autoencoders, GANs, variational autoencoders, deep reinforcement learning, policy gradients, and more. Packt Publishing Ltd, Birmingham, Mumbai, 2018.

Ruben Noortwyck, M.Sc., Wissenschaftlicher Mitarbeiter am Institut für Fördertechnik und Logistik (IFT), Universität Stuttgart
Tel.: +49 (0)711 685 83475
E-Mail: ruben.noortwyck@ift.uni-stuttgart.de

Univ.-Prof. Dr.-Ing. Robert Schulz, Institutsleiter des Instituts für Fördertechnik und Logistik (IFT), Universität Stuttgart
Tel.: +49 (0)711 685 83771
E-Mail: robert.schulz@ift.uni-stuttgart.de

Adresse:

Institut für Fördertechnik und Logistik, Universität Stuttgart, Holzgartenstraße 15 B, D-70174 Stuttgart