Kooperatives Multiagent Reinforcement Learning mit zentralisiertem Deep Q-Learning

Nima Ahmady-Moghaddam (2417523)

Selbstoptimierende Systeme (SOP)
Betreuender Prüfer: Prof. Dr. Stephan Pareigis
Hochschule für Angewandte Wissenschaften (HAW) Hamburg, 20099 Hamburg, DE
Master Informatik, WiSe 2021
nima.ahmady-moghaddam@haw-hamburg.de

Zusammenfassung. Im Multiagent Reinforcement Learning (MARL) wird eine effektive Integration von Methoden des Reinforcement Learning (RL) und Multiagentensystemen (MAS) angestrebt. Mithilfe einer Menge adaptiver und lernender Agenten in einem verteilten intelligenten System sollen komplexe Probleme modelliert und gelöst werden. Durch die Verteilung der Autonomie und Lernfähigkeit auf mehrere Agenten – deren Interessen und Ziele voneinander divergieren können – nimmt die Gesamtkomplexität von MAS zu. In dieser Arbeit wird MARL ausgehend von Single-Agent Reinforcement Learning (SARL) konzeptionell beschrieben. Anschließend werden Eigenschaften des Lernens mehrerer Agenten zur kooperativen Lösung einer gemeinsamen Aufgabe an einem vorimplementierten Modell qualitativ herausgearbeitet. Insbesondere werden betrachtet: (i) die ungleiche Verteilung von Rewards auf Agenten, (ii) die Zustandsrepräsentation im Sinne des Lernens mehrerer Agenten, (iii) die geeignete Verwendung von Endzuständen zur Vermeidung von Exploits durch Agenten. Die Ergebnisse zeigen: (i) Für eine gleichmäßige Aktivitätsverteilung der Agenten eines MAS ist eine Belohnungsstruktur notwendig, die jedem Agenten einen Reward gibt, welcher proportional zu seinem Beitrag zum Gesamtfortschritt der Agenten ist – da sonst einzelne Agenten trotz Inaktivität belohnt werden können. (ii) Die Zustandsrepräsentation, anhand welcher optimales Verhalten gelernt werden soll, muss Eigenschaften der Umgebung, die für die Koordinationsfähigkeit der Agenten wichtig sind, geeignet zusammenfassen, damit ein solches Koordinationsverhalten gelernt werden kann. (iii) Die gezielte Verwendung von Endzuständen kann verhindern, dass mehrere Agenten lernen, in einer unerwünschten Schleife ihre Rewards zu maximieren, ohne das gegebene Problem zu lösen.

Schlüsselwörter: Multiagent Reinforcement Learning · Kooperation · Deep Q-Learning · Stochastic Game · Non-stationarity · Multiagentensystem.

1 Einleitung

Single-Agent Reinforcement Learning (SARL) hat in den letzten Jahren in unterschiedlichen Domänen beachtliche Leistungen und Erfolge verzeichnet. Daher

besteht die Hoffnung, dass Algorithmen und Verfahren des Reinforcement Learning (RL), die sich in SARL-Kontexten bewährt haben, auf Problemstellungen mit mehreren Agenten übertragbar sind [2, 4, 13]. Solche Problemstellungen sind unter dem Sammelbegriff Multiagent Reinforcement Learning (MARL) zusammengefasst. Die Herausforderungen, die einem Multiagentensystem (MAS) innewohnen, sind konzeptionell auf die Komplexitäten von verteilten Systemen zurückzuführen. Durch die Dezentralisierung (d. h., Verteilung) von Intelligenz und Lernfähigkeit entsteht zwar eine erhöhte Parallelisierbarkeit und Leistungsfähigkeit des Systems, aber auch ein erhöhter Koordinations- und Organisationsbedarf. Daher erfordert die Realisierung von MAS mit RL-gestützten Agenten nicht nur die Integration von RL-Ansätzen in einzelne Agenten, sondern auch die Entwicklung von Kommunikationsstrategien zwischen Agenten, die im SARL-Kontext nicht notwendig sind.

Für die vorliegende Arbeit wurden einige der Herausforderungen von MARL auf konzeptioneller und praktischer Ebene an einem einfachen Beispiel [6] betrachtet. Es wurde ein vorimplementiertes Modell mit vortrainierten Agenten in unterschiedlichen Konfigurationen ausgeführt, um gezielt bestimmte Aspekte solcher Systeme zu beleuchten. Das Modell ist ein MAS, dessen Agenten durch ein zentrales Deep Q-Network (DQN) trainiert werden, kooperativ eine Aufgabe optimal zu lösen. Unter anderem wurden bei der Modellausführung Eigenarten der Belohnungsstruktur und Zustandsrepräsentation herausgearbeitet, die qualitativ beschrieben werden.

Die restliche Arbeit ist wie folgt gegliedert. Im Abschnitt 2 sind Grundlagen, die für diesen Themenbereich für wichtig befunden werden, in zusammengefasster Form erarbeitet. Im Abschnitt 3 sind ausgewählte Arbeiten vorgestellt, in denen Deep Q-Learning im Rahmen eines MAS eingesetzt wurde. Im Abschnitt 4 ist das Modell und die Problemstellung beschrieben und die ausprobierten Konfigurationen erläutert. Im Abschnitt 5 sind qualitative Ergebnisse dargestellt. Im Abschnitt 6 sind die Ergebnisse diskutiert und mit den Grundlagen und der Problemstellung des Modells in Verbindung gebracht. Im Abschnitt 7 ist abschließend eine Zusammenfassung der wesentlichen Erkenntnisse enthalten.

2 Grundlagen

In diesem Abschnitt werden Grundlagen aufgearbeitet, die für das Verständnis von MARL und MAS wichtig sind. Es folgt zunächst eine kurze Beschreibung von SARL und dem zugrundeliegenden Markov Decision Process (MDP), um einen allgemeinen Rahmen für die vertiefenden Konzepte zu schaffen. Aufbauend darauf wird das MARL-Problem erläutert und anhand des Stochastic Game formalisiert. Abschließend wird das Verfahren Deep Q-Learning beschrieben, da es im betrachteten Modell für den Lernfortschritt der Agenten zuständig ist.

2.1 Single-Agent Reinforcement Learning (SARL)

RL ist ein Teilbereich des maschinellen Lernens (ML), der sich mit der Optimierung von Handlungen (Actions) von Akteuren (Agents) in Abhängigkeit vom Zu-

stand (State) einer Umgebung (Environment) im Sinne der Maximierung einer unmittelbaren Belohnung (Reward) und einer langfristigen Gesamtbelohnung (Return) beschäftigt.[1] In der Abbildung 1 wird die wechselseitige Interaktion zwischen einem Agent und einem Environment für einen Zeitschritt dargestellt. Wenn der Agent anhand eines RL-Verfahrens lernfähig ist, kann ein solches Setting als SARL bezeichnet werden.

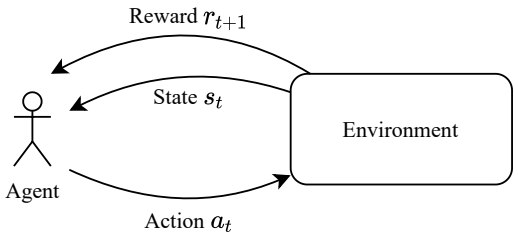

Abb. 1: Darstellung der wechselseitigen Interaktion zwischen einem Agent und seinem Environment für einen Zeitschritt t im Rahmen des SARL. Ein Agent observiert einen State s_t des Environments und führt basierend darauf eine Action a_t aus. Daraufhin erhält der Agent einen Reward r_{t+1}, der die Güte von a_t ausdrückt. (Abbildung angelehnt an [15]).

Der Agent wählt zum Zeitpunkt t in Abhängigkeit des States s_t eine Action a_t aus. Dafür erhält der Agent einen Reward r_{t+1}. Die Durchführung von a_t verändert den State des Environments. Dieser Prozess findet fortlaufend über einen normalerweise diskretisierten Zeitraum statt. Die Summe aller Rewards, die der Agent über einem Zeitschritt t bis zum Ende einer Episode sammelt, wird als Return G_t bezeichnet. Das in der Abbildung 1 dargestellte Wechselspiel kann als Markov Decision Process (MDP) formalisiert werden.

Definition 1. Ein MDP ist ein Tupel $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ mit den folgenden Komponenten:

- $\mathcal{S}$ ist die (endliche) Menge der States
- $\mathcal{A}$ ist die endliche Menge der Actions
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to P(\mathcal{S})$ ist die Zustandsübergangsfunktion (engl. state transition function). $P(\mathcal{S})$ ist eine Wahrscheinlichkeitsverteilung, die für jeden State $s \in \mathcal{S}$ die Übergangswahrscheinlichkeiten in alle möglichen States $s' \in \mathcal{S}$ enthält.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ ist die Belohnungsfunktion (engl. reward function).

[1] In der restlichen Arbeit werden weitestgehend englischsprachige Bezeichnungen verwendet, da diese in der Literatur üblich sind.

– $\gamma \in [0, 1]$ ist der Diskontfaktor (engl. discount factor), der erwartete Rewards zunehmend diskontiert, je weiter diese in der Zukunft liegen[2]

Ein Environment, das als MDP beschrieben werden kann, enthält ausschließlich States, die die Markoveigenschaft[3] erfüllen. In einem solchen Environment kann das Entscheidungsverhalten eines Agents (d. h., die zustandsabhängige Auswahl von Actions) mithilfe von RL-Methoden optimiert werden. Diese Methoden optimieren das Verhalten eines Agents anhand dessen Policy π und Wertefunktion (engl. value function). Die Policy $\pi \colon \mathcal{S} \to P(\mathcal{A})$ eines Agents beschreibt dessen unmittelbares Entscheidungsverhalten, indem es einen State auf eine Wahrscheinlichkeitsverteilung über die möglichen Actions abbildet. Eine Value Function beschreibt den langfristigen Wert eines States für den Agent. Die State-Value Function $v_\pi(s) = \mathbb{E}_\pi[G_t \,|\, S_t = s]$ berechnet hierfür den erwarteten Return G_t unter der Annahme, dass ausgehend vom State s der Policy π gefolgt wird. Die Action-Value Function $q_\pi(s, a) = \mathbb{E}_\pi[G_t \,|\, S_t = s, A_t = a]$ hingegen berechnet den erwarteten Return G_t unter der Annahme, dass ausgehend vom State s die Action a gewählt und dann der Policy π gefolgt wird.

Das Lösen eines MDP besteht darin, eine Policy π zu finden, die $v_\pi(s)$ oder $q_\pi(s, a)$ maximiert. Hierbei sind $\mathcal{P}$ und $\mathcal{R}$ üblicherweise unbekannt und müssen vom Agent durch wiederholtes Ausprobieren gelernt werden. Lösungsansätzen für Probleme dieser Art gehen meistens von zwei grundlegende Annahmen aus: (i) die Zustandsänderungsdynamik des Environment ist unveränderlich (engl. stationary) und (ii) das Environment ist vollständig observierbar (engl. fully observable).[4] Aufgrund dieser zwei Eigenschaften kann ein Agent in vielen SARL-Problemen durch genügend Erfahrung die (eingangs unbekannte) Zustandsübergangsdynamik $\mathcal{P}$ und Belohnungsdynamik $\mathcal{R}$ des Environments erlernen und π optimieren.

2.2 Von SARL zu MARL

Der im Abschnitt 2.1 beschriebene MDP (siehe Definition 1), mithilfe dessen SARL-Probleme modelliert werden können, eignet sich nicht zur Modellierung von MARL-Problemen. Ein Grund hierfür liegt darin, dass in einem MARL-System die optimale Policy π_* für einen Agenten nicht nur von den Gegebenheiten des Environments (insb. $\mathcal{P}$ und $\mathcal{R}$), sondern auch von den Policies der anderen Agenten abhängt [12]. Trotzdem kann aufbauend auf dem Formalismus des MDP und des SARL-Problemkontexts eine Beschreibung von Multiagent Reinforcement Learning (MARL) erarbeitet werden [2]. In der Abbildung 2 wird

[2] Wenn $\gamma \approx 0$, dann werden kurzfristige Rewards priorisiert. Wenn $\gamma \approx 1$, dann werden langfristige Rewards priorisiert.

[3] Ein State $S_t \in \mathcal{S}$ ist Markov $\iff \mathbb{P}[S_{t+1} \,|\, S_t] = \mathbb{P}[S_{t+1} \,|\, S_1, S_2, \ldots, S_t]$

[4] Wenn Full Observability in einem MDP nicht gegeben ist, dann handelt es sich um einen sogenannten Partially Observable MDP (POMDP), bei dem der Agent pro Zeitschritt t eine Beobachtung (engl. Observation) o_t des Environments erhält, die nicht notwendigerweise gleich dem State s_t des Environments ist.

die Erweiterung von SARL nach MARL durch die Hinzunahme mehrerer Agents dargestellt.

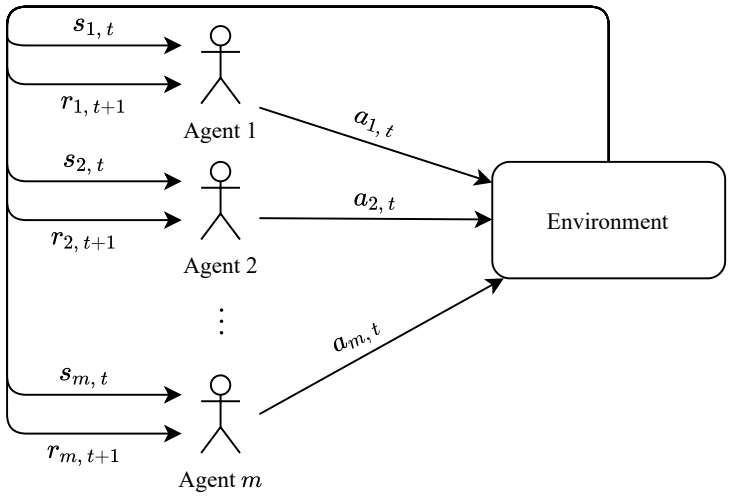

Abb. 2: Darstellung der wechselseitigen Interaktion zwischen m Agents und einem gemeinsamen Environment im Rahmen des MARL. Zum Zeitpunkt t observiert ein Agent i den State $s_{i,t}$ des Environments und führt basierend darauf eine Action $a_{i,t}$ aus. Daraufhin erhält der Agent i einen Reward $r_{i,t+1}$, der die Güte von $a_{i,t}$ ausdrückt. (Abbildung angelehnt an [15]).

Zum Zeitpunkt t observiert jeder Agent i der insgesamt m Agents einen Zustand $s_{i,t}$ des Environments. Dementsprechend führt jeder Agent eine Action $a_{i,t}$ aus und erhält einen eigenen Reward $r_{i,t+1}$. Dieses Interaktionsgefüge zwischen m Agents in einem Environment kann auf unterschiedliche Arten formlisiert werden. Eine Möglichkeit besteht darin, den auf einen Agent beschränkten MDP (siehe Definition 1) für m Agents zu verallgemeinern. Diese Verallgemeinerung wird als Stochastic Game (oder Markov Game) bezeichnet [8, 13].

Definition 2. Ein Stochastic Game ist ein Tupel $(m, \mathcal{S}, \mathcal{A}_i, \mathcal{P}, \mathcal{R}_i, \gamma)$, das einen MDP mit mehr als einem Agent darstellt und sich daher wie folgt von einem MDP unterscheidet:

- $m > 1$ ist die Zahl der beteiligten Agents
- $\mathcal{A}_i$ ist die Menge der Actions des Agents i
- $\mathcal{P} \colon \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times \ldots \times \mathcal{A}_m \to P(\mathcal{S})$ ist die State Transition Function.
- $\mathcal{R}_i \colon \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times \ldots \times \mathcal{A}_m \to \mathbb{R}$ ist die Reward Function des Agents i

Jedem Agent i wird eine Menge von Actions $\mathcal{A}_i$ zugeschrieben. Daraus folgt, dass zwei beliebige Agents i und j möglicherweise unterschiedliche Actions ausführen können (d. h., $\mathcal{A}_i \neq \mathcal{A}_j$). Da in jedem Zeitschritt t eines Stochastic Game

m Agents durch jeweils eine Action $a_{i,t}$ auf den State s_t des Environments einwirken, ergibt sich die State Transition Function $\mathcal{P}$ aus einem Kreuzprodukt von $\mathcal{S}$ und allen $\mathcal{A}_i$. Dementsprechend hat jeder Agent i eine Reward Function $\mathcal{R}_i$, die sich aus dessen aktuellen State $s_{i,t}$ und der Actions A_i aller Agents ergibt.

Im Abschnitt 2.1 wurde erläutert, dass MDPs von Stationarity und Full Observability ausgehen. Ein Stochastic Game enthält aufgrund der Hinzunahme mehrerer Agents – die jeweils autonom handeln und lernen – per Definition eine Non-Stationarity. Aus Sicht eines Agents bedeutet dies, dass sich die Bedingungen, unter denen im Environment Zustandswechsel geschehen, über die Zeit ändern können. Daraus folgt, dass das optimale Verhalten (d. h., die optimale Policy π_*) über die Zeit variieren kann. Dies erschwert den Lernvorgang eines Agents, da er auf ein variables Ziel hinarbeiten muss – was sich im Laufzeitverhalten von Lernalgorithmen niederschlägt [1, 10, 11]. Die Full Observability hingegen ist in einem Stochastic Game gegeben.[5]

Ein Weg, der Problematik der Non-Stationarity entgegenzuwirken, besteht in der Zentralisierung des Lernvorgangs. Dies kann bspw. durch das Hinzufügen eines weiteren Agents realisiert werden, der die Actions der anderen Agents moderiert. In der Abbildung 3 wird die daraus resultierende Verbindungskonstellation zwischen den Agents und dem Environment veranschaulicht.

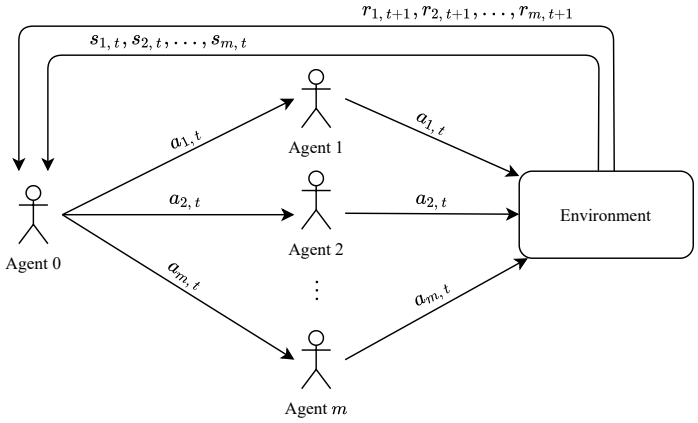

Abb. 3: Darstellung der zentralisierten Optimierung von m Agents durch einen Agent. Der Agent 0 nimmt die State Transitions des Environments entgegen und bestimmt basierend auf ihnen für jeden im Environment aktiven Agent i eine Action $a_{i,t}$. Die daraus folgenden Rewards werden ebenfalls vom Agent 0 entgegengenommen, um die Koordination der Agents zu optimieren.

[5] Wenn Full Observability in einem Stochastic Game nicht gegeben ist, dann handelt es sich um ein sogenanntes Partially Observable Stochastic Game (POSG), bei dem jeder Agent pro Zeitschritt t eine Observation o_t des Environments erhält, die nicht notwendigerweise gleich dem State s_t des Environments ist.

Der Agent 0 erfüllt eine zentrale Empfängerfunktion für Signale aus dem Environment. Er nimmt die Zustände $s_{1,t}, s_{2,t}, \ldots, s_{m,t}$ entgegen und nutzt diese, um für jeden Agent eine entsprechende Action $a_{1,t}, a_{2,t}, \ldots, a_{m,t}$ zu ermitteln. Eine Action $a_{i,t}$ wird vom Agent 0 an den entsprechenden Agent i gegeben, der sie im Environment ausführt. Die sich daraus ergebenden Rewards $r_{1,t+1}, r_{2,t+1}, \ldots, r_{m,t+1}$ werden wiederum vom Agent 0 angenommen.

Durch die Einführung des Agent 0 findet eine Zentralisierung der Lernvorgänge statt, die in einem Stochastic Game normalerweise auf die m Agents verteilt sind. Dadurch werden einerseits sämtliche Fallstricke eines verteilten Systems beseitigt, was die Realisierbarkeit und Beherrschbarkeit des Stochastic Games erhöht. Andererseits ist die Parallelisierbarkeit des Systems dadurch beeinträchtigt, wohingegen sich der Rechenaufwand bei steigendem m exponentiell erhöht. Allgemein muss der zentrale Agent 0 über $|\mathcal{A}_i|^m$ Actions lernen und optimieren, wenn alle Agents i die gleiche Anzahl an Actions durchführen können. Da durch die Zentralisierung diese Rechenlast von einem Agent bewältigt werden muss, skalieren zentralisierte MARL-Systeme mit zunehmender Agentenzahl tendenziell schlecht. Daher ist ein solcher Ansatz nur für relativ kleine Zustandsräume und Agentenzahlen zu empfehlen.

2.3 Deep Q-Learning

Q-Learning ist ein off-policy Temporal Difference (TD) Control-Verfahren, das die optimale Action-Value Function q_* durch eine gelernte Action-Value Function $Q(s, a)$ approximiert [14, S. 131–132]. Hierfür muss während des Lernens gegeben sein, dass alle State-Action Paare fortlaufend besucht werden, damit ihre Q-Values weiterhin angepasst werden können. Dies stellt der Q-Learning Algorithmus durch den Einsatz einer sogenannten ϵ-greedy Strategie sicher, die einen Ausgleich anstrebt zwischen (i) Exploration des Environments und (ii) Exploitation des Wissens über das Environment. Das Verfahren soll $Q(s, a)$ derart optimieren, dass sie gegen q_* konvergiert, wie in der Gleichung 1 beschrieben.

$$q_*(s, a) = \mathbb{E}[r_{t+1} + \gamma \max_{a'} Q(s', a')] \tag{1}$$

Viele der klassischen RL-Verfahren sind tabellarische Lösungsansätze [14, S. 23], d. h., sie verwalten eine Tabelle zum expliziten Abspeichern und Optimieren der einzelnen Werte einer Value Function. Auch Q-Learning kann mithilfe einer Tabelle angewandt werden. Aber bei wachsendem State Space und/oder Action Space ist ein tabellarischer Ansatz rechnerisch ungeeignet. Eine Alternative besteht darin, die zu optimierenden Werte zu approximieren. Da tiefe Künstliche Neuronale Netze (KNN) universelle Funktionsapproximierer sind [5], lassen sie sich zum Approximieren von Value Functions im Rahmen von RL-Problemen einsetzen.

Wenn im Rahmen von Q-Learning KNNs zum Approximieren von q_* verwendet werden, dann spricht man vom Deep Q-Learning. Die verwendeten KNN bezeichnet man als Deep Q-Network (DQN). In der Abbildung 4 wird ein DQN beispielhaft dargestellt. Das DQN nimmt einen State S auf und berechnet für

ihn eine Menge von Q-Values, die jeweils einer Action a_i zugeordnet sind. Das Optimieren dieser Berechnungen erfolgt durch die Anpassung der Gewichte auf den Kanten zwischen den Neuronenschichten.

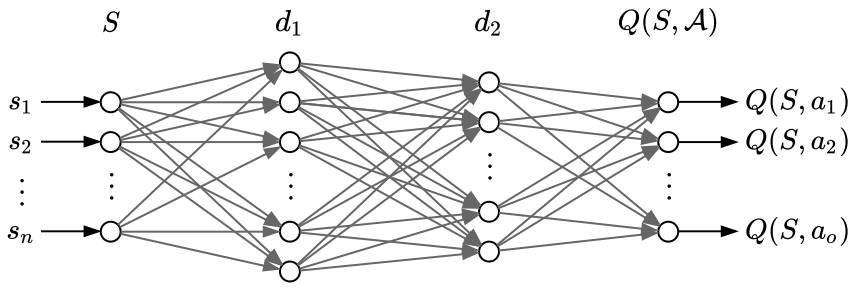

Abb. 4: Darstellung eines DQN mit zwei Schichten d_1 und d_2, das einen State S entgegennimmt und Q-Values aller verfügbaren Actions $a_i \in \mathcal{A}$ ausgibt.

Das Lernverhalten eines DQN lässt sich durch die Verwendung einer vorübergehend statischen Kopie verbessern. Hierzu wird das zu trainierende DQN (Training Network) kopiert und die kopierte Version (Target Network) für eine bestimmte Dauer eingefrohren. Die Gewichte des Target Network werden vom Training Network als q_* erachtet, sodass das Training Network $Q(s, a)$ gegen ein vorübergehend festes Ziel optimieren kann. In regelmäßigen Abständen wird die das Target Network mit den aktuellen Werten des Training Network überschrieben, sodass dieses weiterhin lernen kann.

3 Verwandte Arbeiten

In diesem Abschnitt wird ein Überblick über Forschungsbeiträge gegeben, die sich mit Deep Q-Learning in MAS beschäftigt haben. Hierdurch sollen die im Abschnitt 2 erarbeiteten Grundlagen in einen allgemeinen und zeitgemäßen Kontext eingeordnet werden. Das für diese Arbeit betrachtete MAS verwendet ein zentralisiertes DQN, um Agents geeignetes Verhalten beizubringen. Es gibt einige Forschungsbeiträge, für die Deep Q-Learning in MAS eingesetzt wurde. Aber die meisten Ansätze setzen auf verteiltes Lernen durch mehrere Agents anstelle des in dieser Arbeit thematisierten zentralisierten Lernens durch einen Agenten. Wenngleich dies die Komplexität des Systems aufgrund vieler einzeln lernender Entitäten erhöht, kommt es dem Prinzip von MAS und dem Autonomiegedanken von Agents wesentlich näher als ein zentralisierter Lernansatz. Nachfolgend werden einige Beiträge beschrieben, um das Potential und die Verwendung von DQN in MAS zu veranschaulichen.

Liu et al. (2017) setzten verteiltes Q-Learning in einem MAS zur Entwicklung eines intelligenten Lichtsignalanlagenkontrollsystems ein. Der vorgeschla-

gene Ansatz berücksichtigt die Verkehrsinformationen an benachbarten Kreuzungen sowie den lokalen motorisierten und nicht motorisierten Verkehr, um die Gesamtleistung des Verkehrssteuerungssystems zu verbessern. Ebenfalls wurden Regeln und Einschränkungen des realen Verkehrsgeschehens berücksichtigt. Laut numerischer Experimente trug der Ansatz zu einer Verringerung von, u. a., Wartezeiten von Autos und Fußgängern bei, ohne das allgemeine Sicherheitsrisiko im Verkehr zu erhöhen.

Lin et al. (2018) beschäftigten sich mit der Optimierung von Flottenmanagementsystemen (z. B., für die nachfrageorientierte Allokierung von öffentlichen Transportressourcen in einer Großstadt). Hierzu modellierten sie ein Verkehrsnetzwerk mit Verkehrsteilnehmern als ein MAS, dessen Agents mithilfe von kooperativem Deep Q-Learning eine optimale Ressourcenverteilung erlernen sollten.

Foerster et al. (2017) kombinierten für ihre Studie Elemente des zentralisierten und dezentralisierten Lernens. Im Rahmen eines Actor-Critic Verfahrens wurde ein zentraler Critic zur Approximierung von $Q(s, a)$ und dezentrale Actors zur Optimierung der Policies einzelner Agents eingesetzt.

Yang et al. (2018) machten die allgemeine Beobachtung, dass für kooperative Agenten eines MAS eine Spannung besteht zwischen (i) dem Erreichen individueller Ziele und (ii) dem Beitragen zu kollektiven Zielen. Das Lernen und angemessene Priorisieren beider Ziele ist herausfordernd, weshalb diese Forschungsgruppe probiert hat, die Problemstellung zu sequentialisieren. Demnach sollen Agenten zuerst dem Erlernen persönlicher Zielerfüllung nachkommen und anschließend das Beitragen zum kollektiven Wohl der Agentengruppe in ihren Lernstand integrieren. Dadurch sollen Kooperationsschwierigkeiten beseitigt werden.

4 Modellbeschreibung und Versuchsaufbau

Anhand der im Abschnitt 2 erläuterten Grundlagen wird in diesem Abschnitt das betrachtete Modell beschrieben. Es folgt eine Auflistung der Bestandteile und Regeln des Spiels. Anschließend wird die Reward Function der Agents im Einzelnen erklärt. Zum Schluss werden die verwendeten Parametrisierungen und Konfigurationen zur Ausführung des Modells aufgeführt und motiviert.

4.1 Resource Game: Spielregeln

In der Abbildung 5 ist eine beispielhafte Initialkonfiguration des im Rahmen dieser Arbeit betrachteten MAS dargestellt. Anhand der Abbildung folgt eine Beschreibung der Spieleinheiten und -regeln.

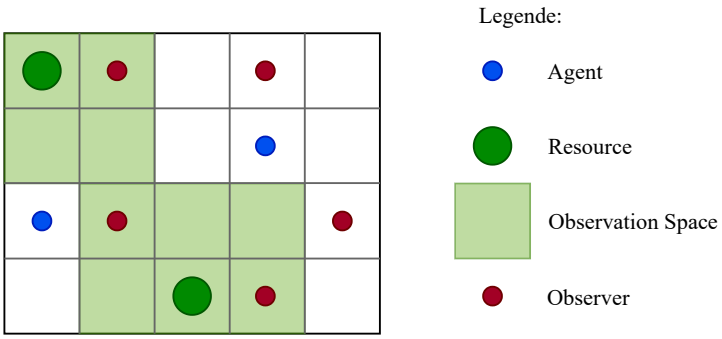

Abb. 5: Beispielhafte Darstellung einer Konfiguration des analysierten Resource Game. Die mobilen Agents (blaue Kreise) haben das Ziel, stationäre Resources (grüne Kreise) einzusammeln, ohne beim Einsammeln von zu vielen stationären Observern (rote Kreise) gesehen zu werden. Die Observer können eine Resource observieren, wenn sie in ihrem Observation Space (grüne Quadrate) sind. Ein Agent kann die Sicht von maximal einem Observer im Observation Space blockieren.

Das Environment besteht aus einer Menge von quadratischen Zellen. Pro Zeitschritt kann sich jeder Agent von der aktuellen Position um eine Zelle in eine der vier Himmelsrichtungen bewegen, oder an der aktuellen Position stehenbleiben. Daraus folgt für alle Agents i, dass $\mathcal{A}_i := \{\texttt{up}, \texttt{down}, \texttt{left}, \texttt{right}, \texttt{idle}\}$. Die Resources, Observer und Observation Spaces sind stationär. Das Ziel der Agents ist es, so viele Resources wie möglich einzusammeln, ohne dabei von zu vielen Observern gesehen zu werden. Ein Agent sammelt eine Resource ein, indem er das Feld der Resource betritt. Ein Observer o kann das Einsammeln einer Resource durch einen Agent beobachten, wenn o im Observation Space der Resource steht. Ein Agent kann die Sicht von o blockieren, indem er den Observation Space betritt, in dem o steht. Daraus folgt, dass zum optimalen Einsammeln einer Resource mindestens genau so viele Agents in ihrem Observation Space anwesend sein müssen Observer. In der Abbildung 5, z. B., müssen zum optimalen Einsammeln der unteren Resource beide Agents im Observation Space sein, um die Sicht der beiden anwesenden Observer zu blockieren. Die Agents müssen daher im Sinne der gemeinsamen Gewinnmaximierung strategisch und insbesondere kooperativ vorgehen.

4.2 Reward Function

Sei $\overline{C}$ die Menge der Resources c_i, A_{c_i} die Menge der Agents, die zum Zeitpunkt des Einsammelns von c_i in ihrem Observation Space sind und O_{c_i} die Menge der Observer, die zum Zeitpunkt des Einsammelns von c_i in ihrem Observation Space stehen. Der Reward r setzt sich aus den folgenden drei Gleichungen zusammen:

$$r(s,a) = r_d + \sum_{i=1}^{|\overline{C}|} f(c_i) \cdot (r_e + r_o) \tag{2}$$

$$f(c_i) = \begin{cases} 1, & \text{wenn } c_i \text{ eingesammelt wurde} \\ 0, & \text{sonst} \end{cases} \tag{3}$$

$$r(o) = \begin{cases} \alpha(p \cdot A_{c_i} - O_{c_i}), & \text{wenn } p \cdot A_{c_i} \leq O_{c_i} \\ \beta(p \cdot A_{c_i} - O_{c_i}), & \text{sonst} \end{cases} \tag{4}$$

Die Gleichung 2 stellt die Reward Function des MAS dar. $r_d \in \mathbb{R}$ ist ein geringfügiger Default-Reward, der pro Zeitschritt – unabhängig davon, ob eine Resource eingesammelt wurde – vergeben wird. r_e ist ein Basis-Reward, der im Fall des Einsammelns einer Resource vergeben wird. r_o und insb. $p \cdot A_{c_i} - O_{c_i}$ stellt die Anzahl der Observer dar, die zum Zeitpunkt des Einsammelns der Resource c_i im ihrem Observation Space aktiv sind (d. h., nicht durch einen ebenfalls anwesenden Agent blockiert werden). Die Gleichung 3 gibt einen Faktor 0 oder 1 zurück, der bestimmt, ob $r_e + r_o$ in die Berechnung des Rewards r einfließt. Die Gleichung 4 soll bewirken, dass Agents (i) entweder für das Einsammeln einer Resource, während mehr oder gleich viele Observer wie Agents anwesend waren, bestraft werden (erster Fall), oder (ii) für das Einsammeln einer Resource, während mehr Agents als Observer anwesend waren, belohnt werden. Diese Dynamik wird von den beiden Skalierungsfaktoren $\alpha, \beta \in \mathbb{R}^+$ reguliert. Hierbei ist $\beta \ll \alpha$, um zu verhindern, dass die Agents sich im Sinne der Maximierung des Rewards auf eine geeignete Resource mit besonders wenigen Observern beschränken und den Rest des Spielfelds ignorieren.

4.3 Versuche und Konfigurationen

Das MAS wurde in drei unterschiedlichen Konfigurationen ausgeführt. In der Tabelle 1 sind die Versuchsparameter aufgeführt.

Tabelle 1: Parametrisierung der durchgeführten Versuche

Versuch	Agents	$\overline{C}$	Observer	p	r_d	r_e	α	β
1	2	2	0	0	-1	10	0	0
2	2	2	5	1	-1	10	6	0
3	2	2	5	1	-1	10	6	0

Der Versuch 1 dient als Basis, um das Bewegungs- und Sammelverhalten der Agents in Abwesenheit von Störungen – Observer und Observation Spaces, durch die das Einsammeln erschwert wird – zu beobachten. Hierfür entfallen p, α und β, da diese nur im dem Observer-Term r_o (siehe Gleichung 4) zum Tragen kommen. In diesem Versuch erscheint für jede eingesammelte Resource eine neue Resource an einer beliebigen Position des Spielfelds. Daher gibt es in

dieser Konfiguration keinen Terminalzustand. Der Versuch 2 bezieht Observer mit ein und setzt dementsprechend Werte für p, α und β. Alle anderen Parameter sind gleich denen des Versuchs 1. Der Versuch 3 ist gleich konfiguriert wie der Versuch 2, hat aber einen Terminalzustand, der dann erreicht ist, wenn alle Resources eingesammelt wurden – d. h., es erscheinen keine neuen Resources mehr, nachdem existierende Resources eingesammelt wurden.

5 Ergebnisse

In diesem Abschnitt werden die qualitativen Ergebnisse präsentiert, die während der Analyse des MAS und der Agents durch die im Abschnitt 4.3 beschriebenen Versuche erhoben wurden. Hierfür sind fünf Videos in das vorliegende Dokument eingebettet, die per Mausklick auf das jeweilige Thumbnail abgespielt werden können. In den Videos wird das Verhalten der Agents unter den jeweiligen Versuchsbedingungen dar. **Anmerkung:** Um ein bereits abgespieltes Video erneut abspielen zu können, kann auf eine andere Seite des Dokuments gesprungen werden und anschließend die Seite mit dem gewünschten Video wieder besucht werden.

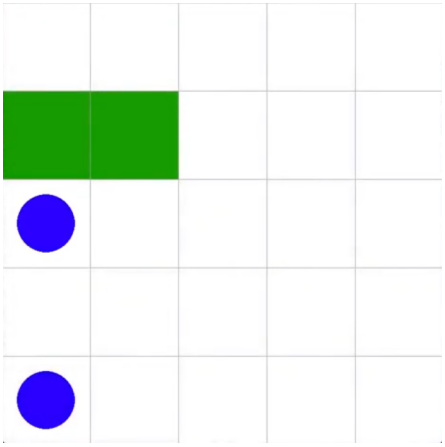

Abb. 6: Versuch 1: jeder Agent erhält unabhängig vom Verhalten den gleichen Reward (Dauer: ca. 30 Sekunden).

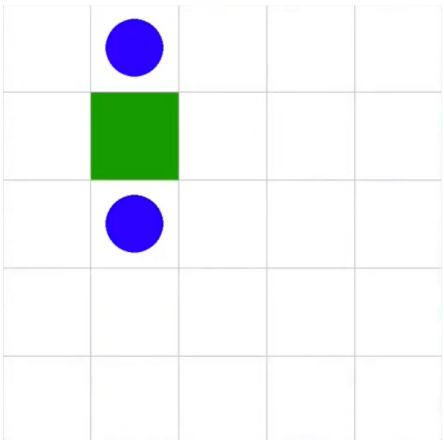

Abb. 7: Versuch 1: jeder Agent erhält einen Reward, der proportional zu seinem Beitrag ist (Dauer: ca. 20 Sekunden).

Im Video 6 wird das Verhalten zweier Agents bei gleichem Reward gezeigt. Jeder Agent erhält einen global definierten Reward, der unabhängig von seinem

Beitrag zum Gesamterfolg der Gruppe (d. h., unabhängig von der direkten Kooperation) ist. Es ist zu sehen, dass ein Agent (hauptsächlich in der linken Hälfte des Spielfelds) wesentlich aktiver ist als der andere Agent (hauptsächlich in der rechten Hälfte des Spielfelds). Im Video 7 ist das Verhalten zweier Agents zu sehen, die jeweils einen Reward erhalten, der proportional zu ihrem Beitrag zum Gesamterfolg der Gruppe ist. Eine solche Definition einer Reward Function pro Agent deckt sich mit dem Formalismus des Stochastic Game (siehe Gleichung 2). Es ist zu sehen, dass in diesem Fall beide Agents aktiv sind.

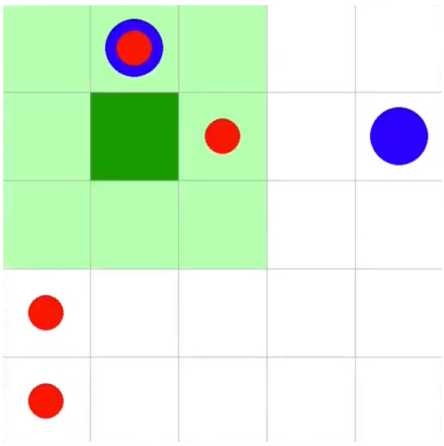

Abb. 8: Versuch 2: expliziter Eingabevektor für das DQN (Dauer: ca. 15 Sekunden).

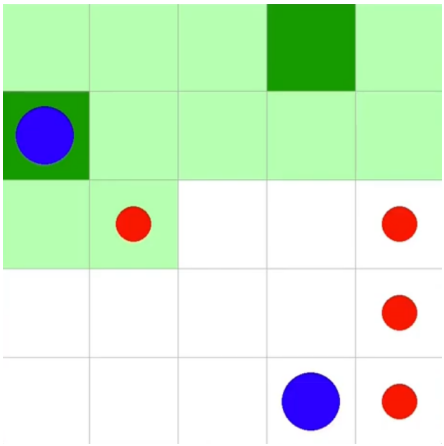

Abb. 9: Versuch 2: vereinfachter Eingabevektor für das DQN (Dauer: ca. 25 Sekunden).

Im Video 8 wird gezeigt, wie sich die Agents bei der Hinzunahme von Observern und Observation Spaces verhalten. Es ist zu sehen, dass die Agents entweder bewegungslos sind oder sich bewegen, aber nicht zielstrebig Resources einsammeln. Dieses Verhalten ist wesentlich anders als das im Video 7 beobachtete Verhalten, in dem noch keine Observer anwesend waren. Für die im Video 9 zu sehenden Ausführung wurde der Eingabevektor des DQN vereinfacht. Ursprünglich wurde dieser Zustandsvektor zusammengesetzt aus den expliziten Positionen (in Form von Koordinaten) der (i) Resources, (ii) Agents und (iii) Observer. Durch die Anpassung des Zustandsvektors wurden anstelle der expliziten Positionen der Observer die Zahl der Observer pro Observation Space jeder Resource angegeben. Dies führt zu dem verhältnismäßig besseren Sammelverhalten der Agents im Video 9.

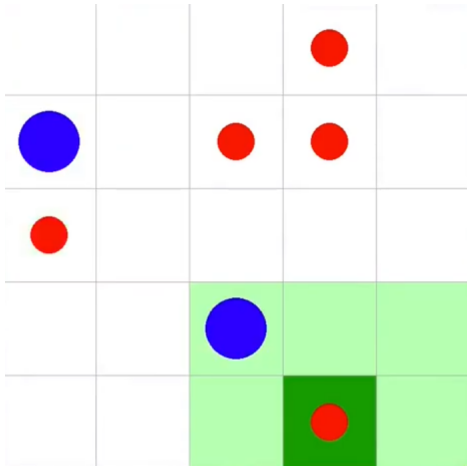

Abb. 10: Versuch 3 (Dauer: ca. 30 Sekunden).

Im Video 10 ist zu sehen, wie sich Agents verhalten, wenn das Erscheinen neuer Resources nach dem Einsammeln existierender Resources deaktiviert ist.

6 Diskussion

In diesem Abschnitt ist eine Analyse der im Abschnitt 5 präsentierten und beschriebenen Ergebnisse enthalten. Es wird auf Besonderheiten des Agentenverhaltens sowie auf die Lösungen für beobachtete Verhaltensprobleme eingegangen.

Die Ergebnisse vom Versuch 1 (siehe Abschnitt 4.3) zeigen, dass mit dem Zentralisieren des Lernens der Agents nicht eine Vereinheitlichung der Rewards der Agents einhergehen darf. Rewards können zwar über eine zentralen Stelle an die jeweiligen Agents verteilt werden (vgl. Abbildung 3). Aber jeder Agent sollte nichtsdestotrotz einen Reward erhalten, der die Güte seines Verhaltens widerspiegelt. Wenn stattdessen, z. B., ein gemittelter Reward oder der Reward eines bestimmten Agents auf alle beteiligten Agents verteilt wird, dann besteht die Möglichkeit, dass Agents unerwünschtes Verhalten lernen. Dies wird als Grund für die beobachteten Verhaltensunterschiede im Video 6 und Video 7 gesehen. Einer der beiden Agents im Video 6 wurde scheinbar mit genügend Rewards belohnt, um keinen Anlass zu sehen, aktiv zu werden und den anderen Agent beim Einsammeln von Resources zu unterstützen.

Der Versuch 2 zeigt, dass die Repräsentation des States für das Lernen optimalen Verhaltens wichtig ist. Im Video 8 hat das DQN als Teil des Zustandsvektors die expliziten Positionen der Observer erhalten. Dies ist zwar im Sinne des Lernens eine sinnvolle Information über den Zustand des Environments. Aber

es stellt keine direkte Verbindung zwischen Observation Spaces und Observern her. Das DQN kann anhand des Zustandsvektors nicht unmittelbar erkennen, welche Observer in welchen Observer Spaces stehen und somit ein Hindernis für die Agents darstellen. Diese Verbindung muss das DQN folglich zusätzlich zur Verhaltensoptimierung erlernen; dies erschwert das Lernproblem unnötig. Die alternative Zustandsrepräsentation, die zum Verhalten im Video 9 geführt hat, gibt für jede Resource die Anzahl der Observer in ihrem Observation Space an. Diese Information ist sowohl kompakter als auch hilfreicher für das DQN, da es durch sie den Zusammenhang zwischen der Zahl der Observer und der dementsprechend benötigten Zahl der Agents für ein optimales Einsammeln der Resource erlernen kann.

Für den Versuch 3 wurde das Erscheinen neuer Resources nach dem Einsammeln existierender Resources deaktiviert. Hierdurch sollte eine eventuelle Auswirkung dieser Eigenschaft des Environments auf das gelernte Verhalten der Agents vermieden werden. Für Agents, denen das Erscheinen neuer Resources bewusst ist, besteht die Möglichkeit, eine schwer einzusammelnde Resource weitestgehend zu ignorieren. Eine Resource ist dann schwer einzusammeln, wenn viele Observer in ihrem Observation Space stehen. Ohne das Erscheinen neuer Resources sind Agents gezwungen, alle vorhandenen Resources einzusammeln, da sie sonst pro Zeitschritt einen negativen Reward (siehe r_d in der Tabelle 1) erhalten und das Spiel nie den Terminalzustand erreicht. Daher werden sie zur Kooperation veranlasst, um selbst schwer einzusammelnde Resources mit möglichst optimalem Reward einzusammeln.

7 Zusammenfassung

In dieser Arbeit wurde ausgehend vom SARL das MARL konzeptionell beschrieben. Eine Formalisierung von MARL wurde anhand des Stochastic Game versucht, welches eine Verallgemeinerung des MDP darstellt (siehe Abschnitt 2.2). Zur praktischen Auseinandersetzung mit diesen Konzepten wurde ein Modell hergenommen, welches vortrainierte Agents zur Verfügung stellt, die in einem gegebenen Environment kooperativ eine Aufgabe lösen müssen (siehe Abschnitt 4). Das Training einzelner Agents erfolgte zentralisiert über ein DQN. Durch diese Zentralisierung sondert sich der für das Modell gewählte Lernansatz von den heutzutage üblichen Ansätzen von MARL (siehe Abschnitt 3) ab. Nichtsdestotrotz konnte das einfache Modell zeigen, dass zentralisiertes Lernen in Environments mit kleinem State Space und Action Space möglich ist und dadurch einige der Koordinationsherausforderungen eines MAS mit verteilter Intelligenz umgangen werden können. Ebenfalls konnten bereits in diesem kleinen Modellrahmen Probleme beim Trainieren mehrerer Agenten herausgearbeitet werden – u. a., (i) die Notwendigkeit einer uneinheitlichen Belohnungsstruktur für die Agents, (ii) die Eignung der Zustandsrepräsentation für das Trainieren der Agents auf koordinatives Verhalten und (iii) die Verwendung von Endzuständen zum Vermeiden von Exploits durch die Agents (siehe Abschnitt 5 und Abschnitt 6).

Eine Möglichkeit, zentralisiertes Lernen in größere MAS zu integrieren, könnte darin bestehen, paarweise disjunkte Teilmengen der Agentenmenge zu bilden und jeder Teilmenge einen zentralen Lernknoten zuzuweisen. Somit könnten Teile des Koordinationsaufwand innerhalb einer einzelner Teilmengen aufgehoben werden. Die Koordinationsproblematik zwischen Teilmengen würde dadurch zwar weiterhin bestehen; aber möglicherweise kann die Gesamtkomplexität eines MAS durch einen solchen dualen Ansatz – zwischen Zentralisierung und Dezentralisierung – verringert werden.

Literaturverzeichnis

[1] Abdoos, M., Mozayani, N., Bazzan, A.L.C.: Traffic light control in non-stationary environments based on multi agent q-learning. In: 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 1580–1585, IEEE (10 2011), ISBN 978-1-4577-2197-7, https://doi.org/10.1109/ITSC.2011.6083114

[2] Canese, L., Cardarilli, G.C., Nunzio, L.D., Fazzolari, R., Giardino, D., Re, M., Spanò, S.: Multi-agent reinforcement learning: A review of challenges and applications. Applied Sciences **11**, 4948 (5 2021), ISSN 2076-3417, https://doi.org/10.3390/app11114948

[3] Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. Association for the Advancement of Artificial Intelligence (AAAI) **24** (5 2017)

[4] Gronauer, S., Diepold, K.: Multi-agent deep reinforcement learning: a survey. Artificial Intelligence Review **55**, 895–943 (2 2022), ISSN 0269-2821, https://doi.org/10.1007/s10462-021-09996-w

[5] Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2**, 359–366 (1 1989), ISSN 08936080, https://doi.org/10.1016/0893-6080(89)90020-8

[6] Köhler, A., Stålvinge, E.G., Furquet, J.R.M.: Coordinated resource collection: Deep q-networks for centralized cooperative multi-agent reinforcement learning. Verfügbar unter: https://jcodingstuff.github.io/docs/NCMML_CoordinatedResourceCollection.pdf (2021), besucht am 04.03.2022

[7] Lin, K., Zhao, R., Xu, Z., Zhou, J.: Efficient large-scale fleet management via multi-agent deep reinforcement learning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1774–1783, ACM (7 2018), ISBN 9781450355520, https://doi.org/10.1145/3219819.3219993

[8] Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of the Eleventh International Conference on Machine Learning, pp. 157–163, Morgan Kaufmann (1994)

[9] Liu, Y., Liu, L., Chen, W.P.: Intelligent traffic light control using distributed multi-agent q learning. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pp. 1–8, IEEE (10 2017), ISBN 978-1-5386-1526-3, https://doi.org/10.1109/ITSC.2017.8317730

[10] Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems (6 2017)

[11] Nguyen, T.T., Nguyen, N.D., Nahavandi, S.: Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. IEEE Transactions on Cybernetics **50**, 3826–3839 (9 2020), ISSN 2168-2267, https://doi.org/10.1109/TCYB.2020.2977374

[12] Nowé, A., Vrancx, P., Hauwere, Y.M.D.: Game Theory and Multi-agent Reinforcement Learning, vol. 12, pp. 441–470. Springer (2012), https://doi.org/10.1007/978-3-642-27645-3_14
[13] Schwartz, H.M.: Multi-Agent Machine Learning: A Reinforcement Approach. Wiley, 1 Aufl. (2014), ISBN 978-1-11-836208-2
[14] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, second Aufl. (2018)
[15] Terry, J.K.: Multi-agent deep reinforcement learning in 13 lines of code using pettingzoo. Verfügbar unter https://towardsdatascience.com/multi-agent-deep-reinforcement-learning-in-15-lines-of-code-using-pettingzoo-e0b963c0820b (2021), besucht am 03.03.2022
[16] Yang, J., Nakhaei, A., Isele, D., Fujimura, K., Zha, H.: Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning. In: International Conference on Learning Representations (ICLR) 2020 Conference (9 2018)

Erklärung zur selbstständigen Bearbeitung einer Hausarbeit

Hiermit bestätige ich, dass ich die vorliegende Arbeit (mit den dazugehörigen Teilen wie Video, Code etc.) selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Die Stellen in der Arbeit, die anderen Werken (Bücher, Artikel, Internetquellen etc.) im Wortlaut, in Übersetzung oder dem Sinn nach entnommen wurden, sind als Zitat mit Angaben der Herkunft deutlich kenntlich gemacht. Dies gilt auch für Abbildungen (Diagramme, Fotos etc.) sowie Programmcodefragmente.

Ich bestätige hiermit außerdem, dass ich weder diese Arbeit noch Teile daraus bereits an anderer Stelle eingereicht habe.

Hamburg, den 6. März 2022

Nima Ahmady-Moghaddam