Karlsruher Institut für Technologie

Fakultät für Informatik
Institut für Anthropomatik
Lehrstuhl für Intelligente Sensor-Aktor-Systeme (ISAS)
Prof. Dr.-Ing. Uwe D. Hanebeck

Intelligente ISAS Sensor-Aktor-Systeme

Diplomarbeit

Dezentrale Regelung eines Multi-Agenten-Systems auf Basis nichtkooperativer Spiele

Markus Kuderer

12. August 2010

Referent: Prof. Dr.-Ing. Uwe D. Hanebeck

Betreuer: Dipl.-Math. Daniel Lyons
Dipl.-Inform. Achim Hekler

Eidesstattliche Erklärung

Hiermit erkläre ich, die vorliegende Diplomarbeit selbstständig angefertigt zu haben. Die verwendeten Quellen sind im Text gekennzeichnet und im Literaturverzeichnis aufgeführt.

Karlsruhe, 12. August 2010

Markus Kuderer

Inhaltsverzeichnis

Abbildungsverzeichnis

Notation

Konventionen

$\mathcal{X}$	Zustandsraum.
$\underline{x}_k$	Gemeinsamer Zustandsvektor zum Zeitpunkt k.
$\underline{x}_k^i$	Zustandsvektor des i-ten Agenten zum Zeitpunkt k.
$\mathcal{U}$	Menge der Aktionen.
$\underline{u}_k$	Gemeinsame Aktion zum Zeitpunkt k.
$\underline{u}_k^i$	Aktion des i-ten Agenten zum Zeitpunkt k.
$\mathcal{Z}$	Beobachtungsraum
$\underline{z}_k^i$	Gemeinsamer Beobachtungsvektor zum Zeitpunkt k.
$\underline{z}_k^i$	Beobachtungsvektor des i-ten Agenten zum Zeitpunkt k.
$\boldsymbol{v}$	Zufallsvariable
$\hat{v}$	Mittelwert einer Zufallsvariablen $\boldsymbol{v}$.
$\mathbf{A}$	Matrix.
$\mathbb{N}$	Menge der natürlichen Zahlen.
$\mathbb{R}$	Menge der reellen Zahlen.
$\sim$	Verteilungsoperator. Z.B. bedeutet $\boldsymbol{v} \sim \mathcal{U}$, dass $\boldsymbol{v}$ gemäß der Verteilung $\mathcal{U}$ verteilt ist.
$\square$	Ende eines Beweises.

KAPITEL 1

Einleitung

1.1 Zusammenfassung

Die dezentrale Regelung von Multi-Agenten-Systemen findet in vielen Bereichen Anwendung. Seien es beispielsweise Roboter-Teams, die eine unbekannte Umgebung erforschen oder Sensornetzwerke, die mit beschränkter Kommunikations- und Rechenleistung Messungen ihrer Umwelt durchführen und dafür ihr Verhalten koordinieren müssen. All diese Systeme beinhalten mehrere autonome Agenten, die in einer gemeinsamen Umgebung agieren, diese beobachten, aber auch aktiv beeinflussen können.

In dieser Arbeit werden Systeme betrachtet, deren Zustand nicht direkt zugänglich ist. Die Agenten können dabei Messungen ausführen, die Rückschlüsse über den Systemzustand zulassen, allerdings kann dieser im Allgemeinen nicht fehlerfrei rekonstruiert werden. Auf Grundlage dieser Information muss in jedem Schritt eine Aktion ausgewählt werden, die als nächstes ausgeführt wird. Die Herausforderung für jeden Agenten besteht darin, zu jedem Zeitpunkt diejenige Aktion auszuwählen, die das beste Ergebnis erwarten lässt. Das gewünschte Verhalten ist dabei über eine Gütefunktion definiert, die zu maximieren ist. Um die Auswirkung von zukünftig gewählten Aktionen vorherzusagen, dient ein stochastisches Systemmodell.

Da aber nicht nur die eigenen Aktionen das Systemverhalten beeinflussen, muss in der Planungsphase zusätzlich das mögliche Verhalten der anderen Agenten berücksichtigt werden. Die Ziele der verschiedenen Agenten können im Allgemeinen auch voneinander abweichen. Somit muss nichtkooperativ geplant werden, da jeder Agent davon ausgehen muss, dass alle anderen Agenten ihre eigenen Ziele verfolgen. Hier wird vorausgesetzt, dass die jeweiligen Ziele global bekannt sind, allerdings keine Kommunikation zwischen den Agenten zugelassen ist. Somit muss jeder Agent dezentral eine Strategie finden, die mit den möglicherweise gewählten Aktionen anderer Agenten vereinbar ist. Ein bekanntes spieltheoretisches Lösungskonzept für Probleme dieser Art sind sogenannte Nash-Gleichgewichte. Ein gemeinsames Verhalten im Nash-Gleichgewicht kann als Kompromisslösung angesehen werden, in welcher die Ziele aller Teilnehmer berücksichtigt werden.

Partially Observable Stochastic Games (*POSG*) modellieren allgemein Systeme dieser Art. Leider ist die Lösung von *POSG*s mit der heute verfügbaren Rechenleistung nachweislich nicht in

kurzer Zeit berechenbar. Werden gewisse Einschränkungen wie Transitions- und Beobachtungsunabhängigkeit vorausgesetzt, so verringert sich die Komplexität deutlich. Doch bestehende Ansätze beschränken sich selbst für diese Systeme meist auf sehr kleine, diskrete Zustandsräume. Viele Probleme, wie zum Beispiel Pfadplanungsprobleme sind aber kontinuierlicher Natur. Ziel ist es daher, ein Verfahren zu entwickeln, mit dem unter gegebenen Einschränkungen Nash-Gleichgewichte über einem kontinuierlichen Zustandsraum effizient berechnet werden können.

Für ein ähnliches eingeschränktes Modell, *Network Distributed-POMDP*, bestehen Verfahren, die Nash-Gleichgewichte mittels alternierender Optimierung berechnen. Dafür wird das vorliegende Problem in lokale Teilprobleme aufgespaltet, die jeweils getrennt gelöst werden. Da dieser Ansatz allerdings auf kleine, diskrete Zustandsräume begrenzt ist, kommt in dieser Arbeit zusätzlich eine approximative Methode für die zentrale Lösung teilweise beobachtbare Systeme zum Einsatz. Dabei wird die Gütefunktion der Teilprobleme punktweise ausgewertet, woraus schnell die optimale Antwort auf veränderte Umgebungsbedingungen gefunden werden kann. Die Kombination dieser Ansätze erlaubt effiziente alternierende Optimierung in kontinuierlichen Zustandsräumen, die zu einer gemeinsamen Strategie konvergiert. Als Anwendungsbeispiel wurde das Verfahren auf Szenarien der Roboter-Pfadplanung angewandt und getestet.

1.2 Motivation

Um die Art der betrachteten Systeme sowie die Problemstellung zu veranschaulichen, zeigt Abb. 1.2 zwei Roboter, die möglichst schnell einen Fluss überqueren wollen. Der Weg an das andere Ufer führt über eine Brücke, die nur Platz für einen Roboter bietet, d.h. die beiden Roboter können die Brücke nicht gleichzeitig benutzen.

Diese Aufgabe ist über eine Funktion definiert, die jeder möglichen Position in dem kontinuierlichen Raum eine Belohnung bzw. Bestrafung zuordnet, die in der Abbildung ebenfalls eingetragen ist. Dementsprechend wird jede Stelle im Wasser bestraft, wobei dem Ziel hingegen ein hoher Wert zugewiesen wird. Zusätzlich ergibt jeder Zustand auf der linken Seite der Brücke einen kleinen negativen Wert, was das Bestreben modelliert, möglichst schnell auf die andere Seite zu wechseln. Jede Kollision wird ebenfalls bestraft. Die Roboter entscheiden nun in diskreten Zeitschritten, welche Bewegung sie als nächstes ausführen. Nach jedem Schritt erhalten sie die der aktuellen Position zugehörige Belohnung. Ziel jedes Roboters ist es, über einen festgelegten, endlichen Zeithorizont möglichst viel Belohnung zu sammeln.

Um Vorhersagen über die Auswirkung bestimmter Aktionen auf zukünftige Zustände treffen zu können, ist den Robotern ein Bewegungsmodell bekannt. Dieses Modell gibt beispielsweise die Schrittlänge bzw. die resultierende Drehung für eine gegebene Aktion an. Diese Größen unterliegen allerdings stochastischem Rauschen, so dass auch der komplette Planungsvorgang stochastisch stattfinden muss. Folglich versuchen die Roboter, den Erwartungswert der Summe aller Belohnungen zu maximieren, die über einen endlichen Horizont zu erreichen sind. Um

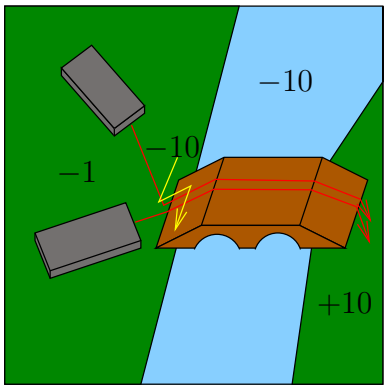


Abbildung 1.1: Beispiel für Multi-Agenten Pfadplanung

mehr Information über den aktuellen Zustand zu erhalten, stehen den Robotern zusätzlich Distanzmessungen zu Landmarken zur Verfügung. Auch diese Messungen sind verrauscht, lassen also nur stochastische Zustandsschätzung zu.

Vor jedem tatsächlich ausgeführten Schritt muss nun jeder Agent seine nächste Aktion wählen. Dieser Vorgang findet dezentral statt, da keine zentrale Planungseinheit vorhanden ist und jeder Roboter seine eigenen Ziele verfolgt. Um optimales Verhalten zu erreichen, müssen zukünftig mögliche Aktionen und Beobachtungen über den kompletten Horizont berücksichtigt werden. Ein Roboter auf der linken Seite des Flusses bestimmt also einen Plan, der ihn möglichst schnell auf die rechte Seite bringt. Nach dieser Planungsphase wird die daraus resultierende nächste Aktion ausgewählt und ausgeführt. Nun stehen durch erneute Messungen weitere Informationen zur Verfügung, die für einen neuen Planungsschritt verwendet werden können.

Wie schon erwähnt, zählt für jeden Roboter nur der eigene Erfolg, also die eigene erhaltene Belohnung. Trotzdem muss auch das Verhalten des anderen berücksichtigt werden, da jede Kollision zusätzliche Bestrafung bedeutet und so dem eigenen Ziel entgegenwirkt. Im dargestellten Beispiel gibt es es für jeden Roboter prinzipiell die zwei Möglichkeiten stehenzubleiben und dem anderen Vortritt zu lassen oder die Brücke zuerst zu überqueren. Insgesamt resultieren daraus vier Kombinationen, von denen zwei für keinen der beiden Roboter vorteilhaft sind. Bewegen sich beide stur auf die Brücke zu, so führt dies zur sicheren Kollision. Warten beide Roboter ab, so können sie zwar nicht kollidieren, allerdings wird die Zielregion nie erreicht. Im Gegensatz dazu sind die zwei verbleibenden Kombinationen, in denen ein Roboter zuerst fährt und der andere wartet, zumindest für einen der Roboter optimal. Der andere kann zwar nicht das theoretisch optimale Ergebnis erreichen, er kann sich aber unter den gegebenen Umständen nicht besser verhalten. In der Spieltheorie wird ein solches Verhalten Nash-Gleichgewicht genannt.

Dieses *rationale* Verhalten ist aber essenziell davon abhängig, dass beiden Robotern bekannt ist, welche gemeinsame Strategie ausgewählt ist. In dieser Arbeit wird vorausgesetzt, dass die Agenten nicht miteinander kommunizieren können. Das bedeutet, die Roboter müssen sich auf Basis der ihnen zur Verfügung stehenden Information für eine gemeinsame Strategie entscheiden und ihren Teil dieser Strategie anwenden. Besitzen die Roboter die gleiche Information über

die Umwelt, so kann durch ein festgelegtes Protokoll garantiert werden, dass alle Roboter zum gleichen Ergebnis kommen und eine rationale gemeinsame Strategie ausführen. Ein solches Verfahren wird in dieser Arbeit vorgestellt.

1.3 Aufbau dieser Arbeit

Zunächst wird formal ein allgemeines Modell eingeführt, das Multi-Agentensysteme mit nicht direkt zugänglichen Zuständen repräsentiert. Anhand dieses Modells wird die Zielsetzung der Regelung definiert und eine kurze Einführung in die Spieltheorie gegeben. Im dritten Kapitel werden bestehende Verfahren für eine optimale oder approximative Lösung dieser Probleme vorgestellt. Kapitel 4 widmet sich den Grundlagen zur Berechnung von optimalen Strategien in teilweise beobachtbaren Umgebungen. Insbesondere wird auf die Transformation in Systeme mit direkt zugänglichem Zustand eingegangen, die mit Hilfe von Standardverfahren gelöst werden können. Diese Betrachtungen bilden die Grundlage für das im Folgenden vorgestellte neuartige Verfahren für die effiziente Regelung von Multi-Agenten-Systemen. Dabei werden zunächst eingeschränkte Voraussetzungen definiert, was eine Trennung des Modells in lokale Teilprobleme erlaubt. Aus den Lösungen dieser Teilprobleme kann mittels alternierender Optimierung eine gemeinsame Strategie berechnet werden. Schließlich wird eine Roboter-Pfadplanungsumgebung beschrieben, anhand derer das Verfahren evaluiert wurde. Die Ergebnisse der durchgeführten Tests werden in Kapitel 6 vorgestellt.

KAPITEL 2

Problemstellung

Das Ziel eines Agenten ist die Maximierung einer Gütefunktion, die das gewünschte Verhalten modelliert. Diese Funktion bezieht sowohl den aktuellen Zustand als auch zukünftige Zustandsübergänge mit ein. Als Stellgröße können die Agenten in jedem Schritt aus einer Menge von verschiedenen Aktionen wählen. Da das unterliegende System- und Messmodell bekannt ist, kann zum Planungszeitpunkt die Auswirkung eines bestimmten zukünftigen Verhaltens auf das System vorhergesagt werden. Dieses Verhalten ist über eine Strategie in jeder Situation definiert. Somit kann das Ziel formal als Optimierungsproblem über dem Raum der möglichen Strategien beschrieben werden.

Da das Modell stochastischer Natur ist, muss der Entscheidungsprozess ebenfalls stochastisch erfolgen, denn die Auswirkung der Aktionen auf zukünftig erreichte Zustände ist nur über bedingte Wahrscheinlichkeiten definiert. In Systemen mit nicht direkt zugänglichen Zuständen sind auch die möglichen Beobachtungen rauschbehaftet. Folglich bestimmt eine Strategie den Erwartungswert der Güte, dessen Maximierung Ziel der Planung ist.

Die Gütefunktion ist im Allgemeinen nicht nur vom eigenen Verhalten abhängig, weshalb auch mögliches Verhalten anderer Agenten in Betracht gezogen werden muss. Diese können eventuell konkurrierende Ziele haben und so Aktionen durchführen, die sich negativ auf die Gütefunktion der restlichen Agenten auswirken. In dieser Arbeit wird ein Spezialfall der dezentralen Planung betrachtet, in dem keine Kommunikation zwischen den Agenten zugelassen ist. Somit muss die Planung lokal von jedem Agenten ausgeführt werden.

2.1 POSG

Ein *Partially Observable Stochastic Game* (*POSG*) modelliert die Interaktion mehrerer Agenten in dynamischen, teilweise beobachtbaren Umgebungen, die jeweils eigene Ziele verfolgen. Formal wird ein *POSG* folgendermaßen beschrieben:

Definition 2.1

Ein *POSG* ist über das Tupel $(\mathcal{X}, \mathcal{U}, T, \mathcal{Z}, O, R^{1,\dots,N})$ definiert.

Dabei ist

- $\mathcal{X}$ der gemeinsame Zustandsraum, $\underline{x} \in \mathcal{X}$ beschreibt den aktuellen Zustand aller Agenten.
- $\mathcal{U} = \mathcal{U}^1 \times \cdots \times \mathcal{U}^N$ die Menge aller gemeinsamen Aktionen. Agent i kann dabei Aktionen aus $\mathcal{U}^i$ wählen.
- $\mathcal{Z} = \mathcal{Z}^1 \times \cdots \times \mathcal{Z}^N$ der Raum der gemeinsamen Beobachtungen. Jeder Agent kann jedoch nur auf seine eigene Beobachtung $\underline{z}^i \in \mathcal{Z}^i$ zugreifen.
- $T(\underline{x}_{k+1}|\underline{x}_k, \underline{u}_k)$ das Systemmodell. Es beschreibt die Transitionsdichte und somit die Auswirkung einer gemeinsamen Aktion auf den gemeinsamen Zustand. Das Modell ist zeitdiskret, d.h. eine Aktion $\underline{u}_k \in \mathcal{U}$ wirkt sich auf den Zustand zum Zeitpunkt $k+1$ aus. Zusätzlich gilt die Markov-Eigenschaft, d.h. $\underline{x}_{k+1}$ ist nur von $\underline{x}_k$ und $\underline{u}_k$ abhängig, nicht von früheren Zuständen und Aktionen.
- $O(\underline{z}_k|\underline{x}_k)$ das Messmodell, definiert die Wahrscheinlichkeit einer Messung $\underline{z}_k$ in gegebenem Zustand $\underline{x}_k$.
- $R^i(\underline{x}_k, \underline{u}_k) \in \mathbb{R}$, mit $i = 1, \ldots, N$ eine Belohnungsfunktion für jeden Agenten. Jede Funktion repräsentiert das gewünschte Verhalten eines Agenten, in dem jeder Kombination aus Zustand und gemeinsamer Aktion eine reelle Zahl zugewiesen wird.

Teilen sich alle Agenten die gleiche Belohnungsfunktion, so spricht man von einem *Decentralized Partially Observable Markov Decision Process* (*Dec-POMDP*). Gibt es eine zentrale Instanz, die Zugriff auf alle Beobachtungen und ausgeführte Aktionen hat, so kann das Problem zentral gelöst werden und wird *Partially Observable Markov Decision Process* (*POMDP*) genannt. Dieser Fall ergibt sich ebenfalls im Spezialfall $N = 1$, wenn also nur ein Agent vorhanden ist.

2.2 Gütefunktion

Die Belohnung für eine Aktion, die durch die Belohnungsfunktion spezifiziert wird, entspricht einer reellen Zahl, die angibt, wie erstrebenswert eine bestimmte Aktion im aktuellen Zustand ist. Plant ein Agent nur für den nächsten Schritt, so sollte er diejenige Aktion wählen, welche maximale Belohnung verspricht. Denn je höher der Wert dieser Funktion, desto besser ist die Aktion für den Agenten zu bewerten. Werden aber nicht nur der aktuelle Schritt, sondern auch weiter in der Zukunft liegende Ereignisse berücksichtigt, so müssen auch mögliche zukünftige Belohnungen mit bewertet werden. Im einleitenden Pfadplanungsszenario wird jeder Aktion, die den Roboter ins Ziel führt, eine hohe Belohnung zugeordnet. Ist das Ziel allerdings nicht mit einem Schritt zu erreichen, so sollten Aktionen ausgewählt werden, die den Roboter weiter in Richtung Ziel führen, selbst wenn diesen Aktionen selbst nur geringe Belohnung zugeordnet ist. Insgesamt ist die zu maximierende Funktion also eine Kombination aus aktueller und möglichen zukünftigen Belohnungen, die als *Gütefunktion* bezeichnet wird.

2.2.1 Endlicher Planungshorizont

Endet der Prozess garantiert nach maximal H Schritten oder wird nur ein endlicher Planungshorizont betrachtet, so kann die kumulative Gütefunktion in natürlicher Weise als Summe aller Einzelbelohnungen definiert werden:

$$R_H(\underline{x}_H) + \sum_{k=0}^{H-1} R(\underline{x}_k, \underline{u}_k),$$

wobei R_H die terminale Belohnung für den letzten Zustand im Planungshorizont angibt. Da zum Planungszeitpunkt die Zustände $\underline{x}_{1,\dots,H}$ jedoch nicht bekannt sind, ist dieser Wert nicht eindeutig definiert. Unter Verwendung des Systemmodells ist es aber möglich, die Verteilungen der zukünftigen Zustände in Abhängigkeit der möglichen Eingabegrößen zu bestimmen. Eine Strategie π beschreibt die Auswahl der Stellgrößen über die nächsten H Schritte. Somit kann der Erwartungswert der erreichbaren Güte für gegebene Strategie berechnet werden:

$$J_\pi = \mathrm{E}\left\{ R_H(\underline{\boldsymbol{x}}_H) + \sum_{k=0}^{H-1} R(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{u}}_k) \right\}. \tag{2.1}$$

Das definierte Ziel eines Agenten ist es also, die erwartete Gütefunktion über endlichem Planungshorizont zu maximieren. In dieser Arbeit wird ausschließlich dieser Ansatz betrachtet, der Vollständigkeit halber werden aber noch zwei weitere weit verbreitete Ansätze vorgestellt, die bei unbeschränktem Planungshorizont Anwendung finden. Diese werden in den folgenden zwei Abschnitten näher erläutert.

2.2.2 Diskontierte Summe

Bei unendlichem Planungshorizont kann der Erwartungswert nicht analog zu einem endlichen Horizont berechnet werden, da die Summe in (2.1) im Allgemeinen nicht konvergiert. Wird aber ein Faktor γ^k eingefügt, um die einzelnen Summanden zu gewichten, so ist bei beschränkter Kostenfunktion R die Konvergenz garantiert. Dieser Ansatz ist als *Diskontierte Summe* bekannt.

$$J_\pi = \mathrm{E}\left\{ \sum_{k=0}^{\infty} \gamma^k R(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{u}}_k) \right\}$$

Zeitlich nähere Ereignisse sind somit im Planungsprozess wichtiger eingestuft als weit in der Zukunft liegende Belohnungen.

2.2.3 Durchschnittliche Belohnung

Ist allerdings die unterschiedliche Gewichtung nicht erwünscht, so gibt es einen weiteren Ansatz, der den Grenzwert der durchschnittlichen Belohnung pro Schritt betrachtet.

$$J_\pi = \lim_{n\to\infty} \frac{1}{n} \mathrm{E}\left\{ \sum_{k=0}^{n} R(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{u}}_k) \right\}$$

Nachteil dieses Verfahrens ist, dass dieser Grenzwert ohne weitere Voraussetzungen im Allgemeinen nicht existiert.

2.3 Strategien

Um die erwartete Gütefunktion zu maximieren, haben die Agenten zu jedem Zeitpunkt die Auswahl zwischen verschiedenen Aktionen. Das Ergebnis des Planungsprozesses in einem Schritt ist somit die nächste auszuführende Aktion. Trotzdem müssen auch mögliche Aktionen in späteren Situationen mitberücksichtigt werden, da diese Einfluss auf die erreichbare Güte haben. Bei endlichem Horizont wirken sich alle Aktionen $\underline{u}_{1,\dots,H}$ auf das erwartete Ergebnis aus, somit muss auch über all diese zukünftigen Entscheidungen optimiert werden. Dafür bietet das stochastische Umweltmodell die Möglichkeit, die Auswirkung zukünftiger Aktionen auf das System vorauszusagen.

Am Anfang dieses Kapitels wurde erwähnt, dass das vorliegende Problem der Optimierung der Gütefunktion über dem Raum aller möglichen Strategien entspricht. Dort wurden Strategien allerdings als „definiertes zukünftiges Verhalten“ eingeführt und nicht weiter spezifiziert. Im Folgenden wird daher detailliert beschrieben, wie Strategien formal repräsentiert werden können.

2.3.1 Einfluss zukünftiger Information

Wird nur über feste Aktionenfolgen optimiert, also zukünftig mögliche Beobachtungen nicht berücksichtigt, so spricht man von *Open Loop Feedback* (*OLF*). In Systemen, bei denen Beobachtungen zur Verfügung stehen, führt diese Art der Planung zu suboptimalen Ergebnissen. Die Berücksichtigung zukünftiger Information kann in diesen Systemen zu besserem Verhalten führen und wird *Closed Loop Feedback* (*CLF*) genannt. Der Unterschied dieser Planungsverfahren soll am Beispiel (Abb. 2.1) verdeutlicht werden.

Im Startzustand $\underline{x}_0$ stehen zwei Aktionen zur Verfügung, die zu unterschiedlichen Folgezuständen führen. Da das Systemverhalten nicht deterministisch ist, soll der Erwartungswert des erreichbaren Gewinns maximiert werden. Die Entscheidung, ob Aktion A oder B ausgewählt werden sollte, hängt nun davon ab, welche Information über den Systemzustand nach Ausführen der ersten Aktion verfügbar sein wird bzw. welches Planungsverfahren angewandt wird.

Wird angenommen, dass nach Ausführung der ersten Aktion keine weitere Information über den Zustand verfügbar ist, müssen „blind“ zukünftige Aktionen gewählt werden. In diesem Fall kann also nur über feste Aktionenfolgen maximiert werden, denn im zweiten Schritt wird nicht

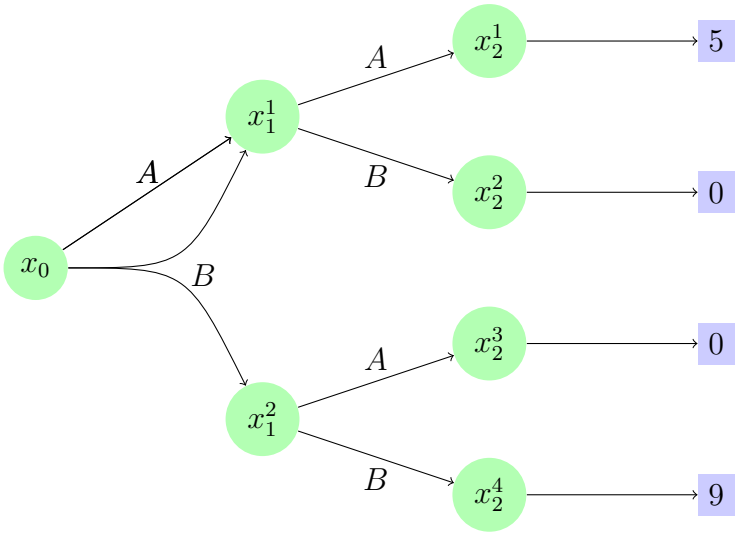


Abbildung 2.1: Beispiel für die Bedeutung zukünftiger Information. Im Startzustand x_0 stehen zwei Aktionen A oder B zur Verfügung. A führt dabei sicher in Zustand x_1^1, B führt mit gleicher Wahrscheinlichkeit in Zustand x_1^1 oder x_1^2. Im zweiten Schritt besteht noch einmal die Wahl zwischen den beiden Aktionen, die je nach erreichtem Zustand zu unterschiedlichen Gewinnen führen.

mehr Information zur Verfügung stehen als zu Beginn. Dieses Verfahren entspricht *OLF*. Der erwartete Gewinn J_π, wobei die Strategie π in diesem Fall einer Abfolge von zwei Zuständen entspricht, ist dann folgendermaßen verteilt:

$$
\begin{aligned}
J_{AA} &= 5 \\
J_{AB} &= 0 \\
J_{BA} &= 0.5 \cdot 5 + 0.5 \cdot 0 = 2.5 \\
J_{BB} &= 0.5 \cdot 0 + 0.5 \cdot 9 = 4.5.
\end{aligned}
$$

Die Strategie AA verspricht offensichtlich den höchsten Gewinn, daher sollte im ersten Schritt Aktion A ausgewählt werden.

Ist aber bekannt, dass der tatsächliche Systemzustand nach dem Ausführen der ersten Aktion zugänglich ist, sollten Strategien verwendet werden, die diese zukünftige Information berücksichtigen. Solche Strategien können als Bäume repräsentiert werden, bei denen jeder Knoten einer Aktion und jede Kante einer Beobachtung entspricht. Die Knoten der Tiefe h geben Aktionen an, die im h-ten Schritt gewählt werden sollten. Jeder Knoten wird dabei über genau eine *Beobachtungshistorie*, d.h. alle Beobachtungen, die zum jeweiligen Zeitpunkt bekannt sind, erreicht. In diesem Beispiel entsprechen die Beobachtungen gerade den Systemzuständen. Abb. 2.2 zeigt einen solchen Strategiebaum, der zukünftige Information mit berücksichtigt. Zunächst wird Aktion B ausgewählt, danach je nach tatsächlicher Beobachtung entweder Aktion A oder B. Diese Strategie führt zu einer erwarteten Güte von $0.5 \cdot 5 + 0.5 \cdot 9 = 7$. Es ist einfach zu

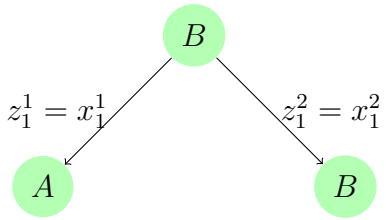


Abbildung 2.2: Strategie im Falle von direkt zugänglichen Zuständen. Die Beobachtungen entsprechen somit gerade den möglichen Zuständen nach dem ersten Schritt 1. Wird Zustand x_1^1 erreicht, so wird durch diese Strategie die Aktion A bestimmt. Wird dagegen x_1^2 erreicht, so wird Aktion B gewählt.

sehen, dass diese Strategie die beste Strategie ist, weshalb im ersten Schritt Aktion B gewählt werden sollte.

Das Beispiel zeigt, dass es wichtig ist, den möglichen Einfluss zukünftiger Information bei der Entscheidung zu beachten, selbst wenn die tatsächliche Instanz dieser Information zum Planungszeitpunkt noch nicht bekannt ist. Hier wurde ein Spezialfall von Systemen betrachtet, in dem der Zustand direkt zugänglich war, d.h. die Beobachtungen gerade den Zuständen entsprachen. In dieser Arbeit werden im Gegensatz dazu allgemeinere Beobachtungen vorausgesetzt, die zwar Rückschlüsse über den Systemzustand zulassen, diesen aber nicht komplett beschreiben.

Im einführenden Beispiel der Pfadplanung können die Roboter verrauschte Distanzmessungen durchführen. Diese Beobachtungen erlauben eine verbesserte Zustandsschätzung, der tatsächliche Zustand kann aber nicht fehlerfrei rekonstruiert werden. Trotzdem kann das Wissen über zukünftige Messungen das Planungsverhalten deutlich beeinflussen. Unterliegt das Bewegungsmodell starkem Rauschen, so würde ein Roboter durch *Open-Loop*-Planung auf der linken Seite des Flusses stehen bleiben. Das Risiko, beim Überqueren der Brücke in den Fluss zu stürzen, wäre in diesem Fall zu groß. Wüsste er allerdings, dass ihm kurz vor der Brücke präzisere Positionsinformationen zukommen, so könnte er sich trotz allem auf die Brücke zubewegen und sein zukünftiges Verhalten je nach tatsächlicher Messung anpassen.

2.3.2 Lokale Strategie

Die Information, die einem Agenten zum Zeitpunkt k zur Verfügung steht, besteht also zusätzlich zur initialen Zustandsverteilung und dem System- und Messmodell aus allen eigenen Aktionen und Beobachtungen. Da diese Information im Allgemeinen relevant für das zukünftige Systemverhalten ist, sollte eine optimale Strategie diese auch berücksichtigen.

Formal ist eine *Ein-Schritt Strategie* des i-ten Agenten also eine Abbildung:

$$\mu_k^i(\underline{z}_0^i, \underline{u}_0^i, \underline{z}_1^i, \ldots, \underline{u}_{k-1}^i, \underline{z}_k^i) \in \Delta\mathcal{U},$$

wobei $\Delta\mathcal{U}$ den Raum der Wahrscheinlichkeitsverteilungen über $\mathcal{U}$ bezeichnet, aus der die tatsächlich ausgeführte Aktion gezogen wird. Bildet eine Strategie statt auf Verteilungen auf einzelne Aktionen ab, so spricht man von einer *reinen* oder *deterministischen Strategie*. Eine komplette Strategie für endlichen Planungshorizont besteht aus allen Einzelstrategien der Schritte 0 bis $H-1$. Eine lokale Strategie ist die Menge aller lokalen Ein-Schritt-Strategien

$$\pi^i = \{\mu_0^i, \dots \mu_{H-1}^i\}.$$

Somit definiert π^i für jeden Zeitpunkt das Verhalten des i-ten Agenten abhängig von den tatsächlich erhaltenen Beobachtungen.

2.3.3 Gemeinsame Strategie

Die Menge aller lokalen Strategien wird *gemeinsame Strategie* genannt und definiert das Verhalten aller Agenten, die Teil des Multi-Agenten-Systems sind:

$$\pi = \pi^1 \times \dots \times \pi^n.$$

Ist eine gemeinsame Strategie π, sowie eine *initiale Zustandsverteilung* $\underline{b}_0$ gegeben, so kann die erwartete Güte über dem Planungshorizont für jeden Agenten bestimmt werden:

$$J_\pi^i(\underline{b}_0) = \mathrm{E}\left\{ R_H^i(\underline{\boldsymbol{x}}_H) + \sum_{k=0}^{H-1} R^i(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{u}}_k)) \Big| \pi, \underline{b}_0 \right\}. \tag{2.2}$$

Die Verteilungsfunktion der Zufallsvariablen $\underline{\boldsymbol{x}}_k$ ist durch $\underline{b}_0$ und π eindeutig bestimmt [Ber95]. Diejenige Strategie, deren Gütefunktion maximal ist, ist die optimale Strategie

$$\pi^*(i) = \arg\max_\pi J_\pi^i(\underline{b}_0). \tag{2.3}$$

Verfolgen die Agenten ein gemeinsames Ziel, so führt dieses Konzept zu einem gemeinsamen optimalen Verhalten. In diesem Fall ist die Belohnungsfunktion aller Agenten gleich, d.h. es gilt auch $J_\pi^i = J_\pi^j$ für alle Agenten i, j. Somit ist auch das Ergebnis der Optimierung (2.3) immer die gleiche gemeinsame Strategie. Es ist damit also ein gemeinsames Verhalten bestimmt, das für alle Agenten die optimale Güte verspricht.

Im Gegensatz dazu ist es oft nicht möglich eine gemeinsame optimale Strategie zu bestimmen, wenn die Agenten unterschiedliche Ziele haben. Dann sind im Allgemeinen auch die Gütefunktionen J_π^i jedes Agenten verschieden. Dies führt zu unterschiedlichen Ergebnissen von (2.3) für jeden Agenten. Jede dieser Lösungen $\pi^*(i)$ definiert aber eine gemeinsame Strategie. Das heißt, die gemeinsame Strategie, die höchste Güte für einen Agenten verspricht, hängt auch vom Verhalten anderer Agenten ab. Diese haben aber im Allgemeinen kein Bestreben, diesen Plan zuzuführen, da für sie andere Strategien höhere Güte versprechen. Im Folgenden wird daher

untersucht, wie rationale, gemeinsame Strategien auch bei konkurrierenden Zielen der Agenten gefunden werden können. Zunächst soll dafür definiert werden, was als *rationales Verhalten* bezeichnet wird.

2.4 Rationales Verhalten

Das im letzten Abschnitt beschriebene Problem ist die zentrale Fragestellung der Spieltheorie. Spieltheorie beschäftigt sich mit dem Verhalten mehrerer Agenten, die sich gegenseitig beeinflussen können und möglicherweise konkurrierende Ziele haben. Dabei versucht jeder Spieler Aktionen auszuführen, die für ihn vorteilhaft sind. Gleichzeitig muss aber auch der Einfluss der möglichen Entscheidungen der Gegenspieler auf den Ausgang beachtet werden. Als Entscheidungsgrundlage gibt es in der Spieltheorie das Konzept des *rationalen Verhaltens*. Jeder Agent geht dabei davon aus, dass sich die anderen Spieler rational verhalten.

Ein Beispiel aus [Web07] verdeutlicht die Idee, die hinter der Definition von rationalem Verhalten steckt. Eine Lotteriegesellschaft bietet zwei verschiedene Lotterien L_1 und L_2 an, wobei die erste einen sicheren Gewinn von einer Million Euro verspricht, die zweite die 50/50-Chance auf drei Millionen Euro. Vor die Wahl gestellt, würden die meisten Menschen wohl die erste Variante wählen, obwohl zweitere einen höheren erwarteten Gewinn verspricht (denn $0.5 \cdot 3 + 0.5 \cdot 0 = 1.5$). Dies zeigt also, dass die einfache Maximierung des erwarteten Gewinns nicht unbedingt rationales Verhalten definiert.

Vielmehr werden *Ausgänge* $\omega \in \Omega$ eines Spiels mit einer Ordnung versehen. Als „Ausgang“ ist dabei eine Wahrscheinlichkeitsverteilung einzelner Ergebnisse gemeint, wie z.B. die 50/50-Chance durch die Wahl von L_2. Von rationalen Spielern wird nun verlangt, alle möglichen Ausgänge gegeneinander in Relation zu setzen. In unserem Beispiel entspricht jede Lotterie selbst einem Ausgang, somit ist $\Omega = \{L_1, L_2\}$ und die typische Ordnung somit

$$L_1 > L_2 \ .$$

Allerdings könnte die Ordnung für einen risikofreudigen Spieler auch umgekehrt ausfallen. Um als *rational* zu gelten, muss eine solche Ordnung aber gewisse Axiome erfüllen, beispielsweise Vollständigkeit und Transitivität. Alle fünf Axiome sowie weitere Details zu diesem Thema sind ebenfalls in [Web07] zu finden. Morgenstern und von Neumann zeigten 1944 in [vNM44], dass es immer eine reelle Funktion $R : \Omega \mapsto \mathbb{R}$ gibt, so dass die Maximierung dieser Funktion gerade dem definierten rationalen Verhalten entspricht. Das bedeutet also, dass der Erwartungswert von R über die Verteilung eines Ausgangs ausgewertet, genau dann größer als der eines anderen Ausgangs ist, wenn dieser auch als „besser“ bezüglich der Ordnung definiert wurde.

Mit dieser Erkenntnis sind die Belohnungsfunktionen eines *POSG* (Def. 2.1) durch reelle Funktion R sinnvoll definiert, da für jede rationale Bewertung der möglichen Ausgänge eine solche Belohnungsfunktion existiert. Jedes gewünschte rationale Verhalten entspricht also der Maximierung einer bestimmten Gütefunktion. Auf Basis dieser Betrachtung kann eines der wichtigsten Konzepte der Spieltheorie, das Nash-Gleichgewicht, eingeführt werden.

2.5 Nash-Gleichgewicht

		Spieler 2	
		Z	S
Spieler 1	Z	$(-2, -2)$	$(0, -5)$
	S	$(-5, 0)$	$(-1, -1)$

Tabelle 2.1: Gefangendilemma: Gewinnmatrix für Spieler 1 und 2. Jedem Spieler stehen die Aktionen Zugeben (Z) und Schweigen (S) zur Verfügung

Das Konzept des Nash-Gleichgewichts wird im Folgenden anhand des Gefangenendilemmas beschrieben. Dieses bekannte Szenario ist in Tabelle 2.1 dargestellt und beschreibt zwei Straftäter, die in getrennten Zellen verhört werden. Beide haben die Wahl, die Tat zuzugeben (Z) oder zu schweigen (S). Schweigen beide, so werden sie aufgrund eines weniger schweren Verbrechens angeklagt und erhalten beide ein Jahr Gefängnis. Geben beide die Tat zu, so müssen sie ihre Strafe von zwei Jahren absitzen. Gibt jedoch einer der beiden die Tat zu, während der andere schweigt, so wird er sofort freigelassen während der andere die Höchststrafe von fünf Jahren erhält.

Eine deterministische lokale Strategie eines Spielers π^i ist in diesem Fall eine der Aktionen Z oder S. Wie in der Tabelle deutlich zu sehen, hängt die erwartete Zeit im Gefängnis aber nicht nur von der eigenen Strategie, sondern auch vom Verhalten des anderen Täters ab. Die gemeinsame Strategie mit global bestem Ausgang (also kleinster gemeinsamer Strafe) wäre für beide zu schweigen, (S, S). In dieser Situation könnte aber jeweils ein Spieler von der gemeinsamen Strategie abweichen, um komplett ohne Strafe auszugehen. Beispielsweise könnte Spieler 1 abweichen und die Tat zugeben. Dann läge die gemeinsame Strategie (Z, S) vor, die für Spieler 1 optimal ist. Die Kombination (S, S) ist also kein stabiles Gleichgewicht, da jeder Spieler einen Vorteil daraus ziehen kann, von der gemeinsamen Strategie abzuweichen. Eine Strategie, die solche Abweichungen nicht erlaubt, wird *Nash-Gleichgewicht* genannt und ist eines der wichtigsten Lösungskonzepte der Spieltheorie.

Definition 2.2 (Nash-Gleichgewicht) Eine gemeinsame Strategie π befindet sich im Nash-Gleichgewicht, wenn für alle Spieler i gilt:

$$R^i(\pi^i, \pi^{-i}) \geq R^i(\tilde{\pi}^i, \pi^{-i}) \text{ für alle } \tilde{\pi}^i.$$

Dabei bezeichnet π^{-i} die Strategie aller Spieler außer i.

Das einzige Nash-Gleichgewicht im Gefangenendilemma ist (Z, Z). Hier entscheidet sich jeder der beiden Straftäter optimal unter der Voraussetzung, dass der andere an der Strategie festhält. Einer der zentralen Sätze der Spieltheorie besagt, dass jedes endliche Spiel mindestens ein solches Nash-Gleichgewicht besitzt, das allerdings nicht eindeutig sein muss [Web07]. In diesem Beispiel wurden nur Strategien über dem Planungshorizont 1 betrachtet, die also nur die nächste Aktion bestimmen. Nash-Gleichgewichte können aber auch auf beliebige Strategien über größerem Horizont angewandt werden, sobald jeder dieser gemeinsamen Strategien eine Güte für jeden Agenten zugewiesen werden kann. Dies ist bei den in dieser Arbeit betrachteten Systemen und zugehörigen Strategien der Fall, wie in Abschnitt 2.3 gezeigt.

KAPITEL 3

Stand der Technik

3.1 Lösungsansätze

Sind $\mathcal{Z}$ und $\mathcal{U}$ endlich, so ist es möglich, alle lokalen Strategien über endlichem Horizont aufzubauen und die zugehörige Gütefunktion zu berechnen. Werden alle Kombinationen dieser lokalen Strategien aufgelistet, so entspricht dies einem Spiel in Normalform, das in jedem Fall ein Nash-Gleichgewicht besitzt. Im Falle einer gemeinsamen Gütefunktion führt die Brute-Force-Maximierung über alle gemeinsamen Strategien sogar zu einer global optimalen Strategie. Dieser Ansatz ist allerdings aufgrund seiner Komplexität eher theoretischer Natur, da es mit der heute verfügbaren Rechenleistung selbst für kleine Beispiele nicht möglich ist, in kurzer Zeit optimale Lösungen zu finden. Im folgenden werden daher bestehende Ansätze vorgestellt, die bestimmte Strukturen in *POSG*s bzw. *Dec-POMDP*s ausnutzen, um diese effizient zu berechnen bzw. approximative Lösungen bestimmen.

Anstatt alle Strategien aufzulisten, durchsucht *Multi Agent* A^* (*MAA**) [SZ05] die Strategiebäume mittels A^*-Baumsuche und entfernt dabei schon bei kleinem Horizont Strategien, die garantiert schlechter als die beste bis dahin bekannte Strategie sind. Um Strategien zu entfernen, muss eine komplette H-Schritt-Strategie als Referenz bekannt sein, sowie eine Heuristik eingesetzt werden, mit der die verbleibende Güte geschätzt wird. Überschätzt diese Heuristik die Güte in jedem Fall, so findet *MAA** garantiert eine optimale Strategie.

Hansen zeigt in [HBZ04], dass *Dynamisches Programmieren* (*DP*) ähnlich wie zur Lösung von *POMDP*s (Kapitel 4.4.1) eingesetzt werden kann. Dafür wird der Raum der Zustandsverteilungen eines Agenten um Verteilungen über die Strategien der anderen Agenten erweitert. So kann die Gütefunktion iterativ aufgebaut werden und gleichzeitig können Strategien entfernt werden, deren Güte an jeder Stelle des erweiterten Zustandsraumes von einer anderen Strategie dominiert wird. Dieses Verfahren entspricht dem Entfernen schwach dominierter Strategien in Normalform. Das Verfahren nutzt die Eigenschaft von *POSG*s über diskretem Zustandsraum, dass die zugehörige Gütefunktion bei endlichem Horizont stückweise linear über dem erweiterten Zustandsraum ist.

Ein anderer Ansatz basiert auf der Transformation eines *POMDP* in eine Folge Bayesscher Spiele, die gleichwertig zum ursprünglichen Problem ist [EMGST04] [EMGST05]. Bayessche

Spiele verallgemeinern Spiele in Normalform insofern, dass Agenten unterschiedliches Wissen über den Zustand der Umgebung besitzen können. Die private Information eines Agenten wird „Typ“ genannt und entspricht in diesem Fall den eigenen Beobachtungen, die den anderen Agenten nicht zugänglich sind. Um die Bayesschen Spiele exakt zu lösen, muss aber in jedem Schritt die erwartete Gütefunktion berechnet werden, die mit der optimalen Strategie zu erreichen wäre. Da diese optimale Strategie berechnet werden soll und somit zu Beginn nicht verfügbar ist, kann diese Transformation nicht direkt für eine effizientere Berechnung eingesetzt werden [OSV08]. Wird aber eine approximierte Gütefunktion verwendet, so können die einzelnen Bayesschen Spiele schnell gelöst werden. Emery-Montemerlo et al. schlägt dafür Q_{MDP} vor [EMGST04], [EMGST05]. Dabei wird angenommen, dass die Zustände in jedem Schritt direkt zugänglich sind. [OSV08] gibt weitere Möglichkeiten an. Zum einen wird Q_{POMDP} vorgeschlagen, wobei eine gedachte zentrale Instanz auf alle Beobachtungen zugreifen kann. Zum anderen kann Q_{BG} verwendet werden, bei dem im Gegensatz zu Q_{POMDP} nicht nur alle gemeinsamen Beobachtungen sondern auch die gewählten Aktionen als bekannt angenommen werden.

Nair et al. schlägt in [NTY+03] den Algorithmus *JESP* vor, der globale Optimalität zu Gunsten von Effizienz opfert, um lokal optimale Lösungen (*Nash-Gleichgewichte*) in *Dec-POMDP*s zu finden. Dabei wird abwechselnd die optimale Strategie eines Agenten berechnet, während die restlichen Strategien festgehalten werden. Das Verfahren konvergiert somit zu einem Nash-Gleichgewicht, da alle Agenten die gleiche Gütefunktion maximieren und diese in jedem Schritt verbessert wird. *DP-JESP* [NTY+03] verwendet Dynamisches Programmieren für die alternierende Optimierung und basiert auf der Idee, dass der *Dec-POMDP* für einen Agenten als *POMDP* mit erweitertem Zustandsraum beschrieben werden kann, solange die Strategien der anderen Agenten festgehalten werden. Der erweiterte Zustand besteht dabei aus einer Kombination von eigenem Zustand und Beobachtungen der anderen Agenten. Während *DP-JESP* Lösungen für bestimmte initiale Zustandsverteilungen liefert, ermöglicht es *CS-POMDP* [VNTY06] die Gütefunktion über dem kompletten erweiterten Zustandsraum vorzuberechnen.

Neben approximativen Lösungsverfahren für *POSG*s bzw. *Dec-POMDP*s gibt es einige Arbeiten, die vereinfachte Probleme betrachten, für die Lösungen effizient bestimmt werden können.

*Network Distributed POMDP*s (*ND-POMDP*) [NVTY05] sind Multi-Agenten-Systeme mit unabhängigen Transitions- und Beobachtungswahrscheinlichkeiten. Das heißt, die Agenten agieren auf separaten Zustandsräumen und interagieren nur über gemeinsame Belohnungsfunktionen. Zusätzlich wird ausgenutzt, dass die Interaktion lokal stattfindet und Agenten nicht gleichzeitig mit allen anderen interagieren. Zur Lösung dieses vereinfachten Modells wird *Locally Interacting Distributed JESP* (*LID-JESP*) beschrieben, das alternierende Optimierung für diese Art von Systemen einsetzt. *LID-JESP* hängt eng mit dem in dieser Diplomarbeit vorgestellten Verfahren zusammen. Varakantham gibt in [VMY+07] weitere Verfahren zur Lösung von *ND-POMDP*s an, die bestimmte Qualitätseigenschaften der erreichten Lösung garantieren.

Allgemeiner beschreibt [OSW08], wie lokale Interaktion in *Dec-POMDP*s ausgenutzt werden kann, ohne Transitions- und Beobachtungsunabhängigkeit vorauszusetzen. Dabei werden Abhängigkeiten des Modells analysiert, um faktorisierte Gütefunktionen aufzustellen. Der *Dec-POMDP* kann so als *Collaborative Graphical Bayesian Game* (*CGBG*) repräsentiert werden.

Ein weiteres Beispiel für eingeschränkte Modelle wird in [GL06] beschrieben. Hier interagieren die Agenten nur, indem sie gegenseitig Aktionen ausschließen. Zusätzlich wird komplette Beobachtbarkeit des eigenen Zustands vorausgesetzt. Der vorgestellte Algorithmus berechnet Worst-Case und Best-Case-Güten und entfernt so iterativ dominierte Strategien.

3.2 Komplexität

Optimale Strategien für Multi-Agenten-Systeme zu finden, ist nachweislich sehr komplex. Insbesondere sind optimale Lösungen nicht effizient berechenbar, wenn der aktuelle Zustand nicht direkt zugänglich ist, sondern Beobachtungen nur unsicherheitsbehaftete Aussagen über den Systemzustand machen. Bernstein zeigt in [BZI00], dass die optimale Lösung von *Dec-POMDP*s, in denen alle Agenten die gleiche Gütefunktion teilen, für endlichen Horizont $NEXP$-vollständig [1] ist. Der allgemeinere Fall, in dem die Agenten unterschiedliche, möglicherweise konkurrierende Ziele haben, erweist sich sogar als noch schwerer lösbar [GM08]. In der Praxis bedeutet das, dass selbst kleinste Beispiele mit wenigen Zuständen nicht mehr in kurzer Zeit zu berechnen sind.

Ein typisches Referenzproblem ist das *Dezentrale Tiger Problem* [NTY+03], bei dem hinter einer von zwei Türen ein hungriger Tiger lauert, während sich hinter der anderen ein Schatz befindet. Die Agenten haben nun jeweils die Wahl zwischen verschiedenen Aktionen. Sie können lauschen, was mit gewisser Wahrscheinlichkeit den Aufenthaltsort des Tigers angibt, allerdings ist diese Aktion mit Kosten verbunden. Zum anderen können sie eine der Türen öffnen. Dabei wird das Finden des Schatzes belohnt, während es für die Wahl der anderen Tür eine Strafe gibt. Selbst die Suche nach lokal optimalen Strategien für zwei Agenten ist beschränkt auf Horizont ~ 7 [NTY+03], was die Komplexität von Problemen dieser Art verdeutlicht.

Auch einige eingeschränkte Modelle, die beispielsweise unabhängige Belohnungsfunktionen voraussetzen, oder die Interaktion nur stattfindet, indem Agenten gegenseitig Aktionen ausschließen [GL06], liegen ebenfalls in der Klasse *NEXP*. Eine Ausnahme bilden Systeme, bei denen die gemeinsamen Beobachtungen den aktuellen Zustand eindeutig identifizieren und zusätzlich Transitions- und Beobachtungsunabhängigkeit vorausgesetzt wird. Diese *Dec-MDP*s sind NP-vollständig [AZ09].

1 NP beschreibt die Klasse der Entscheidungsprobleme, die in polynomieller Zeit von nichtdeterministischen Turingmaschinen entschieden werden können. Probleme der Klasse $NEXP$ können dagegen im gleichen Modell in der Zeit $\mathcal{O}(2^{p(n)})$ entschieden werden.

KAPITEL 4

Grundlagen: Systeme mit nicht direkt zugänglichen Zuständen

Für Systeme mit direkt zugänglichem Zustand lassen sich optimale Strategien mittels Standardverfahren finden, die auf dem Prinzip des *Dynamischen Programmierens* (*DP*) basieren. Dies verringert die Komplexität im Gegensatz zur Brute-Force-Suche über alle Strategien, da das Problem aus Teilproblemen besteht, die sequentiell voneinander abhängen. So können zunächst Teilergebnisse berechnet werden, die zur Lösung der nächsten Stufe beitragen. Dabei wird ausgenutzt, dass der aktuelle Zustand eine *Sufficient Statistic* ist und somit das stochastische Verhalten des Systems komplett beschreibt. Somit kann die frühere Systemhistorie zu jedem Zeitpunkt komplett vernachlässigt werden, da alle relevante Information im aktuellen Zustand steckt, der jeweils direkt zugänglich ist.

Dieses Prinzip lässt sich allerdings nicht direkt auf Systeme mit nicht direkt zugänglichen Zuständen übertragen. Das Problem ist, dass die jeweiligen Beobachtungen den Systemzustand nicht fehlerfrei beschreiben. Zu jedem Zeitpunkt müssen also nicht nur die aktuelle Beobachtung, sondern auch alle früheren Aktionen und Beobachtungen beachtet werden. Ein Modell dieser Art lässt sich jedoch unter gewissen Voraussetzungen in ein komplett beobachtbares System transformieren, dessen Zustandsraum aus dem Raum aller Wahrscheinlichkeitsverteilungen über $\mathcal{X}$ besteht. Die Grundlagen des DP-Prinzips sowie der Zustandstransformation werden nun anhand zentral geregelter Systeme beschrieben.

4.1 POMDP

Zu Beginn dieser Arbeit wurden *POMDP*s als Spezialfall eines *POSG* definiert. Zum besseren Verständnis werden *POMDP*s an dieser Stelle noch einmal gesondert definiert, was aber gleichwertig zur Def. 2.1 im Fall $N = 1$ ist.

Definition 4.1

Ein *POMDP* besteht aus dem Tupel $(\mathcal{X}, \mathcal{U}, T, \mathcal{Z}, O, R)$.

Dabei ist

- $\mathcal{X}$ der Zustandsraum, wobei $\underline{x} \in \mathcal{X}$ den aktuellen Zustand beschreibt,
- $\mathcal{U}$ die Menge der verfügbaren Aktionen,
- $\mathcal{Z}$ der Raum der Beobachtungen,
- $T(\underline{x}_{k+1}|\underline{x}_k, \underline{u}_k)$ das Systemmodell, das die Auswirkung einer Aktion auf den Zustand beschreibt. Auch hier gilt die Markov-Eigenschaft, dass der Zustandsübergang also nicht von Zuständen und Aktionen zu früheren Zeitpunkten abhängt.
- $O(\underline{z}_k|\underline{x}_k)$ das Beobachtungsmodell,
- $R(\underline{x}) \in \mathbb{R}$ die Belohnungsfunktion, mit der die Ziele der Agenten festgelegt sind.

4.2 Direkt zugängliche Zustände

Kann der aktuelle Zustand zum Zeitpunkt k komplett aus der Beobachtung $\underline{z}_k$ rekonstruiert werden, so spricht man von Systemen mit direkt zugänglichen Zuständen. In diesem Fall kann eine Strategie definiert werden, die jedem Zustand eine Aktion zuweist. Neben dem aktuellen Zustand sind zwar auch vorherige Zustände sowie Aktionen bekannt, allerdings kann gezeigt werden, dass diese nicht relevant für das zukünftige Verhalten des Systems sind, was auf die Markov-Eigenschaft zurückzuführen ist. Die Ein-Schritt-Strategien sind somit Abbildungen über dem Zustandsraum:

$$\mu_k(\underline{x}_k) = \underline{u}_k \in \mathcal{U}.$$

Zur besseren Lesbarkeit werden hier nur deterministische Strategien betrachtet, die auf einzelne Aktionen, anstelle von Verteilungen über Aktionen, abbilden. Diese Strategien können als Bäume dargestellt werden, in denen jeder Knoten einer Aktion und jede Kante einem Zustand entspricht (s. Abb. 4.1).

Ist μ_0 bis μ_{H-1} sowie ein Startzustand $\underline{x}_0$ gegeben, so sind auch die Verteilungen des Systemzustands $\underline{x}_k$ für $k = 1, \ldots, H$ eindeutig bestimmt (s. Anhang A.1). Damit kann die erwartete Gütefunktion

$$J_\pi^{\mathrm{MDP}}(\underline{b}_0) = \mathrm{E}\left\{ R_H(\underline{\boldsymbol{x}}_H) + \sum_{k=0}^{H-1} R(\underline{\boldsymbol{x}}_k, \mu_k(\underline{\boldsymbol{x}}_k)) \middle| \pi, \underline{b}_0 \right\},$$

die direkt aus (2.2) folgt, als Summe über alle erwarteten Schrittgüten für jede gegebene Strategie bestimmt werden. Somit kann die Gütefunktion bei endlichem Zustands- und Aktionenraum über alle $(|\mathcal{X}||\mathcal{U}|)^N$ möglichen Strategien maximiert werden [Ber95].

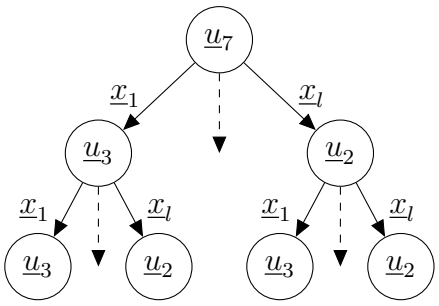


Abbildung 4.1: Bei direkt zugänglichen Zuständen kann eine Strategie als Abbildung über dem Zustandsraum definiert werden. Der Baum ist ein Beispiel für eine Strategie mit festem Startzustand, wobei die Knoten hier festen Aktionen entsprechen, die Kanten entsprechen Zuständen.

4.3 Dynamisches Programmieren

Die Berechnung kann allerdings wesentlich vereinfacht werden, indem die Additivität der Belohnungsfunktion ausgenutzt und das Prinzip *Dynamisches Programmieren* (*DP*) angewandt wird. *DP* basiert auf dem Optimalitätsprinzip von Bellman [Bel57].

Satz 4.1 *Sei $\pi^* = \{\mu_0^*, \ldots, \mu_{H-1}^*\}$ die optimale Strategie eines Systems mit direkt zugänglichen Zuständen mit initialem Zustand $\underline{x}_0$ und sei $\underline{x}_k$ ein Zustand, der zum Zeitpunkt k mit positiver Wahrscheinlichkeit (bezüglich π^*) auftritt. Wird nun das Teilproblem betrachtet, eine optimale Strategie mit Horizont $H - k$ vom Startzustand $\underline{x}_k$ aus zu finden, dann gilt, dass*

$$(\mu_k^*, \ldots, \mu_{H-1}^*)$$

dieses Teilproblem optimal löst.

Diese Aussage ist einfach nachvollziehbar, wenn sie beispielsweise auf die Suche nach dem schnellsten Weg von A nach B angewandt wird. Führt dieser über X, so muss dieser Weg auch zwischen X und B einen kürzesten Weg verwenden. Ansonsten könnte der Weg AB verkürzt werden, in dem ein kürzerer Weg von X nach B gewählt wird. Daraus kann folgender Satz abgeleitet werden [Ber95]:

Satz 4.2 *Für jeden Startzustand $\underline{x}_0$ ist die maximal erreichbare kumulative Gütefunktion $J^*(\underline{x}_0)$ gleich $J_0(\underline{x}_0)$. Dabei wird J_0 gemäß*

$$\begin{aligned} J_H(\underline{x}_H) &= g_H(\underline{x}_H) \\ J_k(\underline{x}_k) &= \max_{u_k} \mathrm{E}_{\underline{x}_{k+1}^{u_k}} \left\{ R(\underline{\boldsymbol{x}}_k, \underline{u}_k) + J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{u_k}) \right\} \end{aligned}$$

rekursiv berechnet, wobei $\underline{x}_{k+1}^{u_k}$ der Zustand ist, der von $\underline{x}_k$ aus mit Aktion $\underline{u}_k$ erreicht wird.

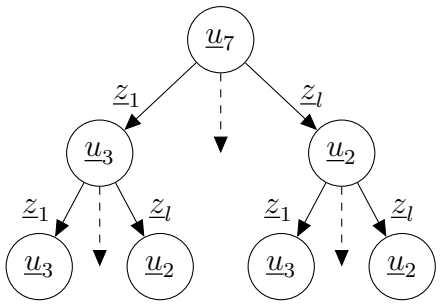


Abbildung 4.2: Bei nicht direkt zugänglichen Zuständen kann eine Strategie als Abbildung über der Systemhistorie definiert werden. Der Baum ist ein Beispiel für eine Strategie mit fester Startzustandsverteilung, wobei die Knoten hier festen Aktionen entsprechen, die Kanten den Beobachtungen zum jeweiligen Zeitpunkt.

Dies ist die direkte, wiederholte Anwendung des Optimalitätsprinzips von Bellman, indem zunächst Teilprobleme der „letzten“ Stufe H betrachtet werden, dann $H-1$ usw. bis J_0 erreicht ist.

4.4 Nicht direkt zugängliche Zustände

Ist es nicht möglich, aus den Beobachtungen den Systemzustand komplett zu rekonstruieren, so können Strategien nicht mehr über dem Zustandsraum definiert werden. Denn um eine solche Strategie anzuwenden, muss zum Zeitpunkt der Ausführung der aktuelle Systemzustand bekannt sein. Dem Agenten steht als Grundlage für seine Entscheidung in diesem Fall die komplette *Systemhistorie*

$$I_k = \{\underline{u}_0, \underline{z}_1, \underline{u}_1, \ldots \underline{u}_{k-1}, \underline{z}_k\}$$

zur Verfügung. Diese enthält alle bisherigen Beobachtungen sowie früher getroffene Entscheidungen. Strategien weisen somit jeder möglichen Abfolge von Beobachtungen Aktionen, bzw. Verteilungen über Aktionen zu. Sind $\mathcal{U}$ und $\mathcal{Z}$ endlich, so können die Strategien wie auch *MDP*-Strategien als Baum dargestellt werden, wie in Abb. 4.2 zu sehen. Die Knoten stellen wieder Aktionen dar, die Kanten entsprechen nun allerdings möglichen Beobachtungen, die nicht die vollständige Rekonstruktion des Zustands zulassen. Zur Ausführungsphase bestimmt dieser Strategiebaum, welche Aktion ausgewählt werden sollte, nachdem eine bestimmte Beobachtung aufgetreten ist.

Vergleicht man die Strategiebäume der Abb. 4.1 und Abb. 4.2, so scheinen die Probleme ähnlicher Natur zu sein. Tatsächlich ist der zweite Fall mit nicht direkt zugänglichen Zuständen aber weitaus komplexer. Der Grund liegt darin, dass die Zustände $\underline{x}_k$ *Sufficient Statistics* des Systems sind, also das zukünftige Verhalten des Systems komplett beschreiben. Ist somit im Strategiebaum Abb. 4.1 die Stufe k erreicht, so ist irrelevant, wie der Zustand $\underline{x}_k$ erreicht wurde, d.h. es muss nur noch der Teilbaum betrachtet werden, der $\underline{x}_k$ folgt. Das heißt aber auch, dass

alle Teilbäume gleichwertig sind, die auf der gleichen Stufe den gleichen Startzustand haben. Diese Eigenschaft wird durch *DP* zur effizienten Berechnung ausgenutzt, indem diese Teilbäume nur einmal berechnet werden. Betrachtet man dagegen einen *POMDP*-Strategiebaum, so sind neben der aktuellen Beobachtung auch alle früheren Beobachtungen und Aktionen von Bedeutung. Deshalb kann *DP* nicht direkt angewandt werden.

Trotzdem ist es wie auch bei direkt zugänglichen Zuständen möglich, die optimale Strategie mittels Brute-Force-Optimierung über alle $(|\mathcal{U}||\mathcal{Z}|)^H$ möglichen Strategiebäume zu berechnen. Denn ist ein kompletter *POMDP*-Strategiebaum gegeben, ist das Verhalten eines Agenten, der diese Strategie befolgt, eindeutig bestimmt. Das heißt, auch der Erwartungswert folgender Gütefunktion

$$J_\pi^{\mathrm{POMDP}}(\underline{b}_0) = \mathrm{E}\left\{ R_H(\underline{\boldsymbol{x}}_H) + \sum_{k=0}^{H-1} R(\underline{\boldsymbol{x}}_k, \mu_k(\underline{\boldsymbol{z}}_0, \underline{\boldsymbol{u}}_0, \ldots, \underline{\boldsymbol{u}}_{k-1}, \underline{\boldsymbol{z}}_k)))\Big| \pi, \underline{b}_0 \right\}$$

ist mit gegebener Startzustandsverteilung $\underline{b}_0$ und Strategie π bestimmt [Ber95].

4.4.1 Belief-State MDP

Da die Maximierung über alle möglichen Strategiebäume allerdings sehr komplex ist, ist es wünschenswert, auch dieses Problem mittels *DP* zu lösen. Dies ist tatsächlich möglich, indem der *POMDP* in einen gleichwertigen *MDP* transformiert wird. Wichtig ist dabei, dass die „neuen“ Zustände *Sufficient Statistics* sind, d.h. alle relevante Information, die über das System vorhanden ist, beinhalten. Es kann gezeigt werden, dass Wahrscheinlichkeitsverteilungen über $\mathcal{X}$ diese Voraussetzung erfüllen [Ber95].

Belief-States, die Zustände des transformierten Systems, beschreiben Verteilungen über dem ursprünglichen Zustandsraum

$$\underline{b}_k(\underline{x}_k) = p(\underline{x}_k|\underline{z}_0, \underline{u}_0, \ldots, \underline{u}_{k-1}, \underline{z}_k) \in \mathcal{B} = \Delta\mathcal{X}.$$

Diese Verteilung hat also den gleichen Informationsgehalt über mögliches zukünftiges Verhalten des Systems wie die komplette Systemhistorie. Das transformierte System wird in der Literatur meist als *Belief-State-MDP* bezeichnet. Die $\underline{b}_k$ können mit gegebenen Beobachtungen wie folgt rekursiv berechnet werden:

$$
\begin{aligned}
\underline{b}_{k+1} &= \phi(\underline{b}_k, \underline{u}_k, \underline{z}_{k+1}) \\
&= p(\underline{x}_{k+1}|I_k, \underline{u}_k, \underline{z}_{k_1}) \\
&= \nu \cdot p(\underline{z}_{k+1}|\underline{x}_{k+1})\ p(\underline{x}_{k+1}|I_k, \underline{u}_k) \\
&= \nu \cdot p(\underline{z}_{k+1}|\underline{x}_{k+1}) \int p(\underline{x}_{k+1}, \underline{x}_k|I_k, \underline{u}_k) d_{\underline{x}_k} \\
&= \nu \cdot p(\underline{z}_{k+1}|\underline{x}_{k+1}) \int p(\underline{x}_{k+1}|\underline{x}_k, \underline{u}_k) p(\underline{x}_k|I_k) d_{\underline{x}_k} \\
&= \nu \cdot p(\underline{z}_{k+1}|\underline{x}_{k+1}) \int p(\underline{x}_{k+1}|\underline{x}_k, \underline{u}_k) \underline{b}_k(\underline{x}_k) d_{\underline{x}_k} \\
&= \nu \cdot O(\underline{z}_{k+1}|\underline{x}_{k+1}) \int T(\underline{x}_{k+1}|\underline{x}_k, \underline{u}_k) \underline{b}_k(\underline{x}_k) d_{\underline{x}_k}, \qquad (4.1)
\end{aligned}
$$

mit ν als konstantem Normierungsfaktor. Somit sind die Systemzustände $\underline{b}_k$ des Belief-State-MDPs direkt zugänglich, da sie komplett über die Beobachtungen rekonstruiert werden können. Die Beobachtungswahrscheinlichkeiten $O(\underline{z}_k|\underline{b}_k)$ ergeben sich durch Integration über den Zustandsraum

$$
\begin{aligned}
O(\underline{z}_k|\underline{b}_k) &= p(\underline{z}_k|\underline{b}_k) \\
&= \int p(\underline{z}_k, \underline{x}_k|\underline{b}_k) d\underline{x}_k \\
&= \int p(\underline{z}_k|\underline{x}_k) p(\underline{x}_k|\underline{b}_k) d\underline{x}_k \\
&= \int O(\underline{z}_k|\underline{x}_k) \underline{b}_k(\underline{x}_k) d\underline{x}_k.
\end{aligned}
$$

Damit kann schließlich auch die Transitionsdichte T berechnet werden, denn es gilt

$$
\begin{aligned}
p(\underline{b}_{k+1}|\underline{b}_k, \underline{u}_k) &= \int_{\mathcal{Z}} p(\underline{b}_{k+1}, \underline{z}_{k+1}|\underline{b}_k, \underline{u}_k) d\underline{z}_{k+1} \\
&= \int_{\mathcal{Z}} p(\underline{b}_{k+1}|\underline{z}_{k+1}, \underline{b}_k, \underline{u}_k) p(\underline{z}_{k+1}|\underline{b}_k, \underline{u}_k) d\underline{z}_{k+1} \\
&= \int_{\mathcal{Z}} p(\underline{b}_{k+1}|\underline{z}_{k+1}, \underline{b}_k, \underline{u}_k) O(\underline{z}_{k+1}|{\underline{b}_k}^{\underline{u}_k}) d\underline{z}_{k+1}. \qquad (4.2)
\end{aligned}
$$

Die erste Multiplikant in (4.2) ist durch (4.1) definiert, ${\underline{b}_k}^{\underline{u}_k}$ bezeichnet die Verteilung, die durch Prädiktion von $\underline{b}_k$ mit Aktion $\underline{u}_k$ entsteht. Zur kompletten Definition des *Belief-State-MDP*s fehlt nun noch eine Belohnungsfunktion, die über $\mathcal{B}$ definiert ist. Eine solche Funktion kann in natürlicher Weise über den Erwartungswert von R definiert werden:

$$R(\underline{b}_k, \underline{u}_k) = \mathrm{E}_{\underline{x}_k}\{R(\underline{\boldsymbol{x}}_k, \underline{u}_k)|\underline{b}_k\}\,.$$

Der *DP*-Operator für *Belief-State MDP*s lässt sich schließlich folgendermaßen darstellen:

$$\begin{aligned} J_H(\underline{b}_H) &= R_H(\underline{b}_H) \\ J_k(\underline{b}_k) &= \max_{\underline{u}_k}\big[R(\underline{b}_k, \underline{u}_k) + \mathrm{E}\big\{J_{k+1}(\underline{\boldsymbol{b}}_{k+1}|\underline{u}_k)\big\}\big] \\ &= \max_{\underline{u}_k}\big[R(\underline{b}_k, a_k) + \int_{\mathcal{Z}}\big\{J_{k+1}(\phi(\underline{b}_k, \underline{u}_k, \underline{z}_{k+1}))p(\underline{z}_{k+1}|\underline{b}_k, \underline{u}_k)\big\}d\underline{z}_k\big]. \end{aligned} \tag{4.3}$$

Die zur optimalen Strategie gehörende Gütefunktion eines so definierten Systems entspricht gerade der optimalen Gütefunktion des ursprünglichen *POMDP*:

$$J_{\pi^*}^{\text{Belief-State MDP}}(\underline{b}_0) = J_{\pi^*}^{\text{POMDP}}(\underline{b}_0).$$

In [Ber95] wird diese Aussage bewiesen. Somit ist es möglich, ein *POMDP* mittels *DP* zu lösen, denn aus gegebener Gütefunktion folgt direkt die optimale Strategie, wie im nächsten Abschnitt gezeigt wird.

4.4.2 Strategie aus Gütefunktion

In Abb. 4.3 ist ein Suchbaum dargestellt, wie er zur Berechnung der optimalen Gütefunktion aufgebaut wird. Um einen Schritt der Berechnung zu beschreiben, wird zunächst der Knoten $\underline{b}_{11}^{\mathrm{p}}$ betrachtet, der aus der initialen Verteilung $\underline{b}_0^{\mathrm{e}}$ mit Aktion $\underline{u}_1$ folgt. Dieser Knoten beschreibt die Zustandsverteilung nach der Prädiktion, also $\underline{b}_{11}^{\mathrm{p}} = p(\underline{x}_1^{\mathrm{p}}|\underline{b}_0^{\mathrm{e}}, \underline{u}_1)$. Zu diesem Zeitpunkt können zwei verschiedene Messungen $\underline{z}_1$ und $\underline{z}_2$ auftreten, die den Baum weiter aufspannen. Durch Filterung der prioren Verteilung mit jeweils einer dieser Beobachtungen entstehen die entsprechenden Zustandsschätzungen $\underline{b}_{11}^{\mathrm{e}}$ sowie $\underline{b}_{12}^{\mathrm{e}}$. Zum Planungszeitpunkt ist zwar noch nicht bekannt, welche der beiden Messungen tatsächlich auftritt, mit Hilfe des Beobachtungsmodells O können aber die Wahrscheinlichkeiten $p(\underline{z}_1|\underline{b}_{11}^{\mathrm{p}})$ und $p(\underline{z}_2|\underline{b}_{11}^{\mathrm{p}})$ bestimmt werden. Ist die Güte $J_1(\underline{b}_{11}^{\mathrm{e}})$ und $J_1(\underline{b}_{12}^{\mathrm{e}})$ für die beiden gefilterten Verteilungen bekannt, kann damit die erwartete Güte der prioren Verteilung $\underline{b}_{11}^{\mathrm{p}}$ berechnet werden.

Auf die gleiche Weise kann die Güte des rechten Knotens $\underline{b}_{12}^{\mathrm{p}}$ berechnet werden, der aus der Prädiktion der initialen Verteilung mit Aktion $\underline{u}_2$ folgt. Somit ist beiden prioren Verteilungen $\underline{b}_{11}^{\mathrm{p}}$ und $\underline{b}_{12}^{\mathrm{p}}$ eine Güte zugewiesen. Die optimale Aktion entspricht schließlich der Aktion, die zur höher bewerteten Verteilung führt.

Da die Güten für $\underline{b}_{11}^{\mathrm{e}} \ldots \underline{b}_{14}^{\mathrm{e}}$ aber im Gegensatz zur eben getroffenen Annahme nicht bekannt sind, muss an diesen Knoten die Berechnung rekursiv weitergeführt werden. Die Rekursion

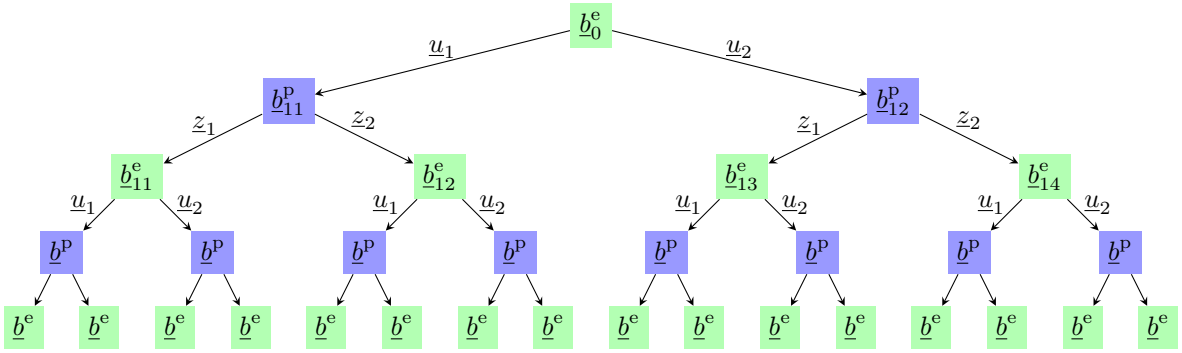


Abbildung 4.3: POMDP-Suchbaum. Die Wurzel des Baumes stellt die Dichte der initialen Zustandsverteilung $\underline{b}_0^{\mathrm{e}}$ dar. In jedem Schritt stehen zwei Aktionen zur Verfügung, die zur prädizierten Verteilung $\underline{b}^{\mathrm{p}}$ führen. Das Beobachtungsmodell liefert ebenfalls zwei mögliche Messungen, die jeweils zu verschiedenen gefilterten Zustandsschätzungen $\underline{b}^{\mathrm{e}}$ führen. Insgesamt ist der *DP*-Vorgang bis zur Tiefe $H = 2$ dargestellt.

endet bei Stufe H, da J_H der erwarteten Belohnungsfunktion R_H entspricht. Der Suchbaum muss also zunächst komplett aufgebaut werden, um anschließend rückwärts die Gütefunktion durch abwechselnde Erwartungswertbildung und Maximierung zu bestimmen.

Alternativ ist es unter bestimmten Voraussetzungen möglich, die Gütefunktion über dem kompletten Zustandsraum $\mathcal{B}$ offline vorzuberechnen, um in der Planungsphase nicht mehr den ganzen Baum aufbauen zu müssen. Dazu wird zunächst $J_H(\underline{b}_H)$ über ganz $\mathcal{B}$ mit der Belohnungsfunktion initialisiert. Im zweiten Schritt wird J_{H-1} über $\mathcal{B}$ berechnet, was möglich ist, da zu diesem Zeitpunkt die Güte J_H bestimmt ist. Dieser Vorgang wird bis zur Gütefunktion J_1 wiederholt. Im Prinzip werden somit alle Suchbäume für alle möglichen initialen Verteilungen gleichzeitig berechnet. Dabei wird ausgenutzt, dass *Belief-States*, die in verschiedenen Bäumen auftauchen, nur einmal berechnet werden müssen. Die Berechnung ist in dieser Form offensichtlich nur möglich, wenn die Gütefunktion geschlossen darstellbar ist oder $\mathcal{B}$ diskretisiert werden kann. Zweiteres wird für die effiziente Berechnung des in Kapitel 5.7 beschriebenen Verfahrens ausgenutzt. In Folge dieser Vorberechnung ist der Aufwand zur Bestimmung der optimalen Aktion nicht mehr abhängig vom Planungshorizont, denn für gegebenes $\underline{b}_0$ muss nur die erste Stufe des oben beschriebenen Verfahrens ausgeführt werden, da die Gütefunktion $J_1(\underline{b}_1)$ direkt für jede Dichte ausgewertet werden kann.

KAPITEL 5

Effiziente Multi-Agenten-Regelung auf Basis lokaler Gütefunktionen

Die Komplexitätsbetrachtung in Kapitel 3.2 zeigt, dass sich die optimale Regelung eines Multi-Agenten-Systems schon für kleinste Beispiele nicht mehr effizient auszuführen lässt. Viele Anwendungen, darunter auch Pfadplanungsprobleme, können allerdings mit vereinfachten Modellen repräsentiert werden, die weitaus einfacher zu lösen sind. Nair et al. beschreibt das eingeschränkte Modell *Network Distributed POMDP*, in dem die stochastischen Prozesse der einzelnen Agenten nur über gemeinsame Gütefunktionen gekoppelt sind [NVTY05]. Um Strategien für Systeme dieser Art zu berechnen, wird unter anderem *LID-JESP* beschrieben, ein Algorithmus, der auf alternierender Optimierung basiert und lokal optimale Lösungen findet. Trotz der deutlich geringeren Komplexität sind die betrachteten Beispielsysteme noch immer auf einen kleinen, diskreten Zustandsraum beschränkt.

Im folgenden Kapitel wird daher ein Verfahren vorgestellt, das ebenfalls auf alternierender Optimierung basiert. Dabei wird ausgenutzt, dass sich das System aufgrund der Transitions- und Beobachtungsunabhängigkeit in schwach gekoppelte Teilprobleme unterteilen lässt, deren Lösung die Grundlage für gemeinsame Strategien ist. Diese lokalen Probleme werden im Gegensatz zu *LID-JESP* mit punktbasierten Verfahren approximativ, aber effizienter gelöst. Die Gütefunktionen dieser lokalen Probleme werden dafür an diskreten Stellen ausgewertet, woraus schnell die optimale Aktion für beliebige Zustände bestimmt werden kann. Ziel des Verfahrens ist es, einen lokal optimalen gemeinsamen Plan zu berechnen, der aus festen Aktionenfolgen für jeden einzelnen Agenten besteht. Über eine parametrisierte Repräsentation der vorkommenden Zustandsverteilungen ist es möglich, berechnungsintensive Komponenten vorzuberechnen, um die Teillösungen in der Planungsphase schnell auf veränderte Bedingungen anzupassen. Auf dieser Grundlage erlaubt alternierende Optimierung die effiziente Suche nach einer lokal optimalen Strategie, d.h. einem Nash-Gleichgewicht, in dem kein Agent das Bestreben hat, von dem gemeinsamen Plan abzuweichen.

5.1 Modell

5.1.1 Transitionsunabhängigkeit

Zunächst wird das zu Grunde liegende Systemmodell formal beschrieben. Transitionsunabhängige Systeme haben die Eigenschaft, dass sich der gemeinsame Zustandsraum in Teilräume für jeden Agenten faktorisieren lässt. Ein Systemzustand besteht also aus einem Vektor, in dem die einzelnen Einträge jeweils dem aktuellen Zustand eines Agenten entsprechen, also

$$\underline{x}_k = (\underline{x}_k^1, \ldots, \underline{x}_k^N) \in \mathcal{X}^1 \times \cdots \times \mathcal{X}^N,$$

wobei $\mathcal{X}^i$ den Zustandsraum des $i-$ten Agenten bezeichnet. Jeder Agent agiert dabei nur auf seinem eigenen Zustandsraum. Das heißt, das Verhalten eines Agenten ist nur von seinem eigenen Zustand abhängig, nicht von den restlichen Einträgen des Zustandsvektors. Zusätzlich gilt, dass auch die Aktionen der anderen Agenten den Folgezustand nicht beeinflussen können. Formal ist die Transitionsdichte T somit das Produkt aller einzelnen Übergangswahrscheinlichkeiten.

$$T(\underline{x}_{k+1}|\underline{x}_k, \underline{u}_k) = \prod_{i=1,\ldots,N} p(\underline{x}_{k+1}^i|\underline{x}_k^i, \underline{u}_k^i)$$

Es wird deutlich, dass die stochastischen Prozesse komplett entkoppelt sind, was die Komplexität der Planung deutlich reduziert. Die Zustandsverteilung eines Agenten nach H Schritten ist ausschließlich von den Aktionen $\underline{u}_{0,\ldots,H-1}^i$ abhängig. Umgekehrt haben diese Aktionen auch keinen Einfluss auf die Zustandsverteilungen der Agenten $j \neq i$.

Transitionsunabhängigkeit ist eine starke Einschränkung des ursprünglichen Modells und angewandt auf physikalische Vorgänge oft nicht realistisch. Meist steht aber nicht die Auswirkung der Interaktion auf den Zustand im Vordergrund, sondern die Auswirkung auf das verfolgte Ziel der Planung. Beim einleitenden Beispiel der gemeinsamen Pfadplanung interagieren die Roboter nur im Falle einer Kollision. Um den Vorgang korrekt zu beschreiben, müssten die daraus folgenden Auswirkungen wie Verformungen bzw. Verschiebung der Roboter im Modell berücksichtigt werden. Für die Planung reicht es allerdings aus, Interaktion durch die Belohnungsfunktion als negativ zu bewerten. So werden in der Ausführungsphase bei korrekter Modellierung Kollisionen vermieden, ohne die Details zu kennen, die daraus resultieren würden. Das heißt, ein allgemeines System wird unter Umständen gut durch ein transitionsunabhängiges Modell beschrieben, in dem die Interaktion durch die gemeinsame Belohnungsfunktion ausreichend repräsentiert wird. Somit findet auch dieses stark eingeschränkte Modell durchaus Anwendung in vielen interessanten Problemen.

5.1.2 Beobachtungsunabhängigkeit

In einem System mit faktorisiertem Zustandsraum kann in ähnlicher Weise Beobachtungsunabhängigkeit definiert werden. Dabei sind die jeweiligen Beobachtungen eines Agenten nur abhängig vom eigenen Zustand und nicht vom Zustand der anderen Agenten. Das stochastische Beobachtungsmodell kann auch hier durch das Produkt der einzelnen Komponenten beschrieben werden.

$$O(\underline{z}_k|\underline{x}_k) = \prod_i p(\underline{z}_k^i|\underline{x}_k^i)$$

Die Information, die jeder Agent über die anderen Agenten besitzt, steckt somit ausschließlich in der initialen gemeinsamen Zustandsverteilung. Da Strategien nur von den eigenen zukünftigen Beobachtungen abhängen, sind sie somit ebenfalls unabhängig vom Zustand der anderen Agenten. Auch dies ist eine Einschränkung des *POSG*-Modells, die aber gleichzeitig die Komplexität erheblich reduziert.

Ist ein System nicht beobachtungsunabhängig, wird es aber in der Planungsphase durch die Annahme modelliert, so entspricht dies einer *Open-Loop*-Approximation der optimalen Strategie (Abschnitt 2.3). Trotzdem können Beobachtungen über den Zustand der anderen Agenten durchaus verwendet werden, indem nach jeder tatsächlich ausgeführten Aktion die gemeinsame Zustandsschätzung entsprechend aktualisiert wird. Einzig in der Planungsphase bleiben somit zukünftig mögliche Messungen unberücksichtigt. Auf die Auswirkungen dieser Approximation wird in Abschnitt 5.8 weiter eingegangen.

In einem transitions- und beobachtungsunabhängigen System sind die einzelnen Agenten also stochastisch komplett entkoppelt. Insbesondere können keine Abhängigkeiten zwischen den Zustandsschätzungen der Agenten entstehen. Ist also die initiale Verteilung des gemeinsamen Systemzustands unabhängig, so werden auch alle weiteren Verteilungen unabhängig sein, was im Folgenden ausgenutzt wird, um den lokalen Prozess als eigenständigen *POMDP* darzustellen.

5.1.3 Additive Belohnungsfunktion

In dem hier vorgeschlagenen Modell besitzt jeder Agent eine eigene lokale Gütefunktion R_L^i, die nur vom eigenen Zustand abhängig ist und daher bei vorausgesetzter Transitionsunabhängigkeit auch nicht von den anderen Agenten beeinflusst werden kann. Die Kopplung zwischen den einzelnen Agenten wird nur durch eine gemeinsame, globale Belohnungsfunktion R_G modelliert, die vom gemeinsamen Zustandsraum abhängt und die von allen Agenten geteilt wird. Somit beeinflussen die Agenten indirekt auch das Verhalten ihrer „Mitspieler“, denn die Belohnungsfunktion ist nun abhängig davon, in welchem Gesamt-Zustand sich das System befindet. Die Belohnungsfunktion des i-ten Agenten besteht somit aus der Summe

$$R^i(\underline{x}) = R^i_L(\underline{x}^i) + R_G(\underline{x}).$$

Folglich setzt sich die von Agent i zu maximierende Gütefunktion folgendermaßen zusammen:

$$J^i(\underline{b}_0) = J^i_L(\underline{b}^i_0) + J_G(\underline{b}_0) = \mathrm{E}\left\{\sum_{k=0}^{H-1}(R^i_L(\underline{\boldsymbol{x}}^i_k, \underline{u}_k))\right\} + \mathrm{E}\left\{\sum_{k=0}^{H-1}(R_G(\underline{\boldsymbol{x}}_k, \underline{u}_k))\right\}.$$

Die initiale Zustandsverteilung des i-ten Agenten wird von $\underline{b}^i_0 = p(\underline{\boldsymbol{x}}^i_0|\underline{b}_0)$ beschrieben. Zugunsten besserer Lesbarkeit wird hier auf die terminale Schrittgüte zum Zeitpunkt H verzichtet (2.1).

5.1.4 ND-POMDPs

Wie schon zuvor erwähnt, zeigt das hier vorgestellte Modell Parallelen zu *ND-POMDP*s, die in [NVTY05] beschrieben werden. Dieses vereinfachte *POSG*-Modell setzt ebenfalls Transitions- und Beobachtungsunabhängigkeit sowie eine additive Belohnungsfunktion voraus. Anstatt lokaler Belohnungsfunktionen für die einzelnen Agenten sowie einer globalen Belohnungsfunktion, die von allen Agenten geteilt wird, können dort durch einen sogenannten *Interaction Hypergraph* auch Gruppen von Agenten definiert werden, die mitereinander interagieren. Für *ND-POMDP*s besteht ein Lösungsverfahren, das auf alternierender Optimierung basiert [NVTY05]. Dabei wird das lokale Problem eines Agenten als *POMDP* über einem erweitertem Zustandsraum dargestellt, der auch die *Beobachtungshistorie* anderer Agenten beinhaltet. Die Gütefunktion dieses *POMDP* ist bei diskreten Systemen stückweise linear und kann somit geschlossen berechnet werden.

Der zentrale Unterschied zum in dieser Arbeit vorgestellten Ansatz liegt in der Repräsentation und Lösung dieser lokalen Probleme. Zunächst wird hier im Gegensatz zu *ND-POMDP*s ein kontinuierlicher Zustandsraum betrachtet. Um dennoch effiziente Regelung zu ermöglichen, werden in dieser Arbeit statt kompletten *POMDP*-Strategien feste Aktionenfolgen betrachtet. Wird vorausgesetzt, dass die initiale Zustandsverteilung unabhängig ist, kann das lokale Problem somit als *POMDP* über dem Zustandsraum eines Agenten dargestellt werden, wie in Abschnitt 5.3 ausführlich beschrieben. Insbesondere durch den Einsatz eines punktbasierten Verfahrens kann eine approximierte Gütefunktion dieses Teilproblems schnell berechnet werden. Dies ist die Grundlage für effiziente alternierende Optimierung über kontinuierlichen, großen Zustandsräumen. Im folgenden Abschnitt wird das Ziel der Planung ausführlich beschrieben.

5.2 Gemeinsame Nash-Gleichgewichte

Das in dieser Arbeit verwendete Modell beschreibt Agenten, die jeweils eigene Ziele verfolgen. Daher kann nicht davon ausgegangen werden, dass ein Agent einem bestimmten Protokoll folgt,

		Roboter 2: A	Roboter 2: B
Roboter 1	A	$(-5, -4)$	$(5, 0)$
	B	$(1, 6)$	$(1, 0)$

Tabelle 5.1: Beispielszenario: Gewinnmatrix für Roboter 1 und 2. Jedem Roboter stehen zwei Strategien zur Verfügung: (A) sofort fahren, oder (B) einige Zeit warten um dann die Brücke zu überqueren.

wenn es für ihn bessere Entscheidungsmöglichkeiten gibt. Ziel des vorgestellten Verfahrens ist es daher, in jedem Schritt eine gemeinsame Strategie zu berechnen, die auch von jedem Agenten ausgeführt wird. Dafür muss sichergestellt werden, dass sich jede Abweichung von dem gemeinsamen Plan negativ auf die Gütefunktion des jeweiligen Agenten auswirkt. Strategien im Nash-Gleichgewicht besitzen diese Eigenschaft, wie in Abschnitt 2.5 gezeigt.

An dieser Stelle wird das Pfadplanungsszenario aus dem ersten Kapitel noch einmal aufgegriffen, um ein Beispiel für das zuvor definierte Modell zu geben und die Problematik gemeinsamer Lösungen darzustellen. In Abb 1.2 sind zwei Roboter zu sehen, die jeweils möglichst schnell eine Brücke überqueren wollen. Vereinfacht dargestellt hat jeder Roboter die Wahl, entweder zuerst die Brücke zu überqueren (Aktion A) oder einige Zeit zu warten, um dann die Brücke zu überqueren (Aktion B). Eine lokale Belohnungsfunktion R_L^1 bzw. R_L^2 modelliert das angestrebte Verhalten. Diese lokalen Belohnungsfunktionen sind nur von der eigenen Aktion des jeweiligen Roboters abhängig und weisen den Aktionen des ersten Roboters die Werte $R_L^1(A) = 5$ und $R_L^1(B) = 1$ zu. Für den zweiten Agenten gilt eine leicht abgeänderte Funktion der Form $R_L^2(A) = 6$ und $R_L^2(B) = 0$.

Zusätzlich gibt es eine gemeinsame Gütefunktion R_G, die beide Roboter in gleichem Maße für Kollisionen bestraft. Somit gilt $R_G(A, A) = -10$, alle anderen Kombinationen von Aktionen werden nicht bestraft. Da die Roboter sich gegenseitig nicht sehen können (Beobachtungsunabhängigkeit), müssen sie sich basierend auf der initialen Verteilung für eine der beiden Varianten entscheiden. Tabelle 5.1 zeigt die gesamte Güte, die für verschiedene Kombinationen von Strategien resultiert.

Die beiden Strategien, die intuitiv als rational empfunden werden, nämlich (A, B) und (B, A), sind in diesem Fall auch Nash-Gleichgewichte. Beide Strategien setzen aber voraus, dass auch der andere Agent die gleiche Strategie wählt, da ansonsten schlechte Ergebnisse für beide Roboter resultieren. Jeweils eine der beiden Varianten wird von einem Agenten bevorzugt, da dies für ihn die optimale Strategie darstellt. Offensichtlich würde Roboter 1 gerne (A, B) anwenden, Roboter 2 favorisiert (B, A). Die Kombination dieser „optimalen“ Strategien, also (A, A), würde aber zur sicheren Kollision führen, was beide Roboter vermeiden wollen. Ein bestimmtes Nash-Gleichgewicht ist somit nur dann sinnvoll anzuwenden, wenn auch alle Beteiligten von seiner Existenz wissen und sicher sind, dass alle anderen diese Strategie verfolgen. Um dies sicherzustellen, gibt es grundsätzlich mehrere Möglichkeiten [KSV05]:

1. Kommunikation
2. Lernen
3. Soziale Konventionen

Ist Kommunikation erlaubt, so bieten sich Auktionen (z.B. [YLB09]) an. Auktionen sind ein Teilgebiet der klassischen Spieltheorie, in dem verschiedene Verhandlungsmethoden betrachtet werden, mit deren Hilfe sich Agenten auf gemeinsame Strategien einigen. Da in dieser Arbeit aber keine Kommunikation während der Planung erlaubt sein soll, wird auf dieses Thema hier nicht weiter eingegangen. Bietet das System für die Agenten die Möglichkeit, wiederholt miteinander zu interagieren, so helfen frühere Erfahrungen, gemeinsame Gleichgewichte zu finden. So könnte z.B. Roboter 1 aus Erfahrung wissen, dass Roboter 2 sich grundsätzlich rücksichtslos verhält und ihm daher in zukünftigen Entscheidungen den Vortritt lassen, um keine Kollision zu riskieren.

Soziale Konventionen sind festgelegte Verhaltensregeln, z.B. bestimmte Ordnungen auf den Nash-Gleichgewichten. Dies könnten z.B. die maximale Summe der lokalen Güten der Agenten sein, was im Beispiel 5.1 eindeutig Strategie (B, A) auswählen würde. Der Ansatz, der in dieser Arbeit gewählt wurde, ist hingegen die feste Priorisierung der Agenten. Die alternierende Optimierung wird in einer bestimmten Reihenfolge der Agenten ausgeführt, was ein eindeutiges Nash-Gleichgewicht garantiert.

Ist mit einem der obigen Ansätze eine bestimmte Strategie im Nash-Gleichgewicht ausgewählt, so ist zu bemerken, dass kein Agent das Bestreben hat, von dieser Vereinbarung abzuweichen, solange er voraussetzt, dass alle anderen Agenten sie befolgen. Dies folgt direkt aus der Definition des Nash-Gleichgewichts, da jede lokale Strategie eines Agenten in diesem Fall optimal in Bezug auf die Strategien der anderen Agenten ist.

Im Gegensatz zu dem eben vorgestellten Beispiel kommen in dieser Arbeit Strategien über einem größeren Horizont zum Einsatz. Um die effiziente Regelung zu ermöglichen, werden allerdings nur feste Abfolgen von Aktionen anstatt kompletter *POMDP*-Strategien betrachtet. Diese Aktionenfolgen werden daher in den folgenden Abschnitten als *Strategie* bezeichnet.

5.3 Lokale Lösung

In der Spieltheorie wird die beste Strategie, die ein Agent ausführen kann, als *beste Antwort* bezeichnet, wenn die Strategien π^{-i} der anderen Agenten bekannt und fest sind. Dieses Konzept ist die Kernidee der alternierenden Optimierung, in der jeweils ein Agent seine *beste Antwort* bestimmt, während die Strategien der anderen Agenten festgehalten werden. Die Lösung eines solchen Optimierungsschrittes wird im folgenden als *lokale Lösung* bezeichnet. Die zu maximierende Gütefunktion hängt dabei nur noch von der eigenen Strategie π^i ab, da die Strategien π^{-i} als fest vorausgesetzt werden:

$$J^i_{\pi^i}(\underline{b}_0) = \mathrm{E}\left\{ \sum_{k=0}^{H-1} R^i(\underline{\boldsymbol{x}}_k, \underline{u}^i_k, \underline{u}^{j\neq i}_k)) \middle| \pi^i, \pi^{-i}, \underline{b}_0 \right\}. \tag{5.1}$$

In [VNTY06] wird gezeigt, dass dieses Problem für allgemeine *POSG* der Lösung eines *POMDP* mit erweitertem Zustandsraum entspricht. Genauer gesagt, wird der Zustand um die *Beobachtungshistorie* der anderen Agenten erweitert.

5.3.1 Lokale Gütefunktion

Mit den zusätzlichen Einschränkungen, die in dieser Arbeit angenommen werden, kann jedoch eine noch einfachere Repräsentation des lokalen Problems gefunden werden. Zum einen sind die fixierten Strategien nicht von Beobachtungen abhängig, sondern feste Aktionenfolgen, d.h. das Verhalten der Agenten $j \neq i$ ist deterministisch festgelegt. Zum anderen sind die stochastischen Prozesse getrennt und nur über die globale Belohnungsfunktion R_G gekoppelt. Ist insbesondere die initiale Zustandsschätzung unabhängig, also $p(\underline{x}_0) = p(\underline{x}^1_0) \cdot \ldots \cdot p(\underline{x}^N_0)$, so bleibt diese Unabhängigkeit auch nach beliebig vielen Prädiktions- und Filterschritten bestehen (s. Anhang A.2). Der Erwartungswert in (5.1)

$$\mathrm{E}\big\{ R^i(\underline{\boldsymbol{x}}_k, \underline{u}_k) \big\} = \mathrm{E}\big\{ R^i_L(\underline{\boldsymbol{x}}^i_k, \underline{u}_k) \big\} + \mathrm{E}\{ R_G(\underline{\boldsymbol{x}}_k, \underline{u}_k) \}$$

ist zwar noch abhängig von den Verteilungen der Zustandsvariablen $\underline{\boldsymbol{x}}^{j\neq i}_k$, allerdings sind diese nicht abhängig von der lokalen Strategie und somit ebenfalls fest. Die erwartete globale Gütefunktion kann also unter den gegebenen Voraussetzungen als Funktion über dem lokalen Zustandsraum beschrieben werden.

$$\begin{aligned}
\mathrm{E}\{R_G(\underline{\boldsymbol{x}}_k, \underline{u}_k)\} &= \int R_G(\underline{x}^1_k, \ldots, \underline{x}^N_k, \underline{u}^1_k, \ldots, \underline{u}^N_k) p(\underline{x}^1_k, \ldots, \underline{x}^N_k | \underline{b}_k) d\underline{x}_k \\
&= \int R_G(\underline{x}^1_k, \ldots, \underline{x}^N_k, \underline{u}^1_k, \ldots, \underline{u}^N_k) \prod_{j=1,\ldots,N} p(\underline{x}^j_k | \underline{b}^j_k) d\underline{x}_k \\
&= \int p^i_k(\underline{x}^i_k | \underline{b}^i_k) \underbrace{\int \prod_{j\neq i} p(\underline{x}^j_k | \underline{b}^j_k) R_G(\underline{x}^1_k, \ldots, \underline{x}^N_k, \underline{u}^1_k, \ldots, \underline{u}^N_k) d\underline{x}^{j\neq i}_k}_{:= R^i_G(\underline{x}^i_k, \underline{u}_k)} d\underline{x}^i_k \\
&= \mathrm{E}\big\{ R^i_G(\underline{\boldsymbol{x}}^i_k, \underline{u}_k) \big\}
\end{aligned} \tag{5.2}$$

Die neu definierte Belohungsfunktion R^i_G bezeichnet dabei die globale Belohnungsfunktion unter der Voraussetzung, dass die Strategien π^{-i} bekannt sind. Somit kann R^i_G im Gegensatz zu R_G als Funktion über dem lokalen Zustand $\underline{x}^i_k$ definiert werden. Dies macht den Weg frei für die folgende Definition eines lokalen *POMDP*, da nun alle Elemente der *POMDP*-Definition in einer lokalen Form dargestellt werden können.

Definition 5.1 (Lokaler POMDP)

Ein lokaler *POMDP* ist ein Tupel $POMDP^i = (\mathcal{X}^i, \mathcal{U}^i, T^i, \mathcal{Z}^i, O^i, R^i)$,

mit

- $\mathcal{X}^i$ Zustandsraum des i-ten Agenten.
- $\mathcal{U}^i$ Aktionen, die Agent i zur Verfügung stehen.
- $T^i(\underline{x}^i_{k+1}|\underline{x}^i_k, \underline{u}^i_k) = p(\underline{x}^i_{k+1}|\underline{x}^i_k, \underline{u}^i_k)$ lokale Transitionsdichte.
- $\mathcal{Z}^i$ Beobachtungen, die nur von $\underline{x}^i \in \mathcal{X}^i$ abhängen.
- $O^i(\underline{z}^i_k|\underline{x}^i_k) = p(\underline{z}^i_k|\underline{x}^i_k)$ lokale Beobachtungswahrscheinlichkeiten.
- $R^i = R^i_L + R^i_G$ wie in (5.2) beschrieben.

Bis auf die Belohnungsfunktion R^i ergibt sich diese Definition direkt aus dem transitions- und beobachtungsunabhängigen Modell, das zu Beginn dieses Kapitels vorgestellt wurde. Mit gegebenen Strategien aller anderen Agenten $\pi^{j\neq i}$ kann auch R^G lokal dargestellt werden, was die Definition des lokalen $POMDP^i$ erlaubt.

In Kapitel 4.4.1 wurde beschrieben, wie für *POMDP*s optimale Strategien gefunden werden können. Diese Strategien sind aber abhängig von zukünftigen Beobachtungen. Somit ist zwar die nächste optimale Aktion $\underline{u}^i_0$ definiert, allerdings nicht die weiteren Aktionen $\underline{u}^i_{1,\dots,H-1}$, wie für das hier vorgestellte Verfahren gefordert. Im folgenden Abschnitt werden daher verschiedene Ansätze beschrieben, wie feste Aktionenfolgen generiert werden können.

5.3.2 Feste Aktionenfolgen

Eine komplette *POMDP*-Strategie weist jeder Sequenz von Beobachtungen $\underline{z}_{1,\dots,k}$ bestimmte Aktionen zu. Abb. 5.1 zeigt ein Beispiel für einen solchen *POMDP*-Strategiebaum eines Systems mit zwei möglichen Beobachtungen und fester initialer Zustandsverteilung. Die erste Aktion ist eindeutig definiert, alle weiteren Aktionen sind aber davon abhängig, ob in folgenden Beobachtungen $\underline{z}_1$ oder $\underline{z}_2$ auftritt. Jede dieser Beobachtungsabfolgen entspricht somit einem bestimmten Weg durch den Baum, wie im dargestellten Beispiel hervorgehoben. Jeder dieser Wege entspricht wiederum einer bestimmten Abfolge von Aktionen. Welche Aktionenfolge zur Ausführungszeit tatsächlich auftritt, ist aber zum Planungszeitpunkt noch nicht bekannt.

Im Gegensatz dazu gibt es approximative Ansätze für die Lösung von *POMDP*s, die eine feste Abfolge von Aktionen generieren. Diese Verfahren wählen in jedem Schritt entweder bestimmte Messungen aus (z.B. *Nominal Belief-State-Optimization* [MHC08]) oder vernachlässigen Messungen sogar komplett (Open Loop Feedback, [YCG+05]). Diese beiden Ansätze werden in folgenden Abschnitten näher beschrieben.

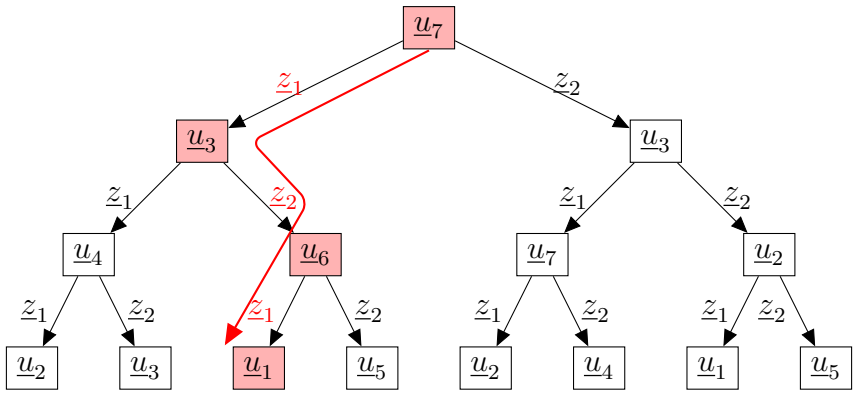


Abbildung 5.1: Beispiel einer *POMDP* Strategie mit $|\mathcal{Z}| = 2$. Zusätzlich ist ein Weg durch den Baum hervorgehoben, der einer festen Abfolge von Aktionen entspricht

Open Loop Feedback

In Abschnitt 2.3 wurde *Open Loop Feedback* (*OLF*) als Optimierung über vereinfachte Strategien vorgestellt. Diese Strategien vernachlässigen zukünftig mögliche Beobachtungen im Planungsvorgang. Die Komplexität verringert sich dadurch drastisch, allerdings kann die approximierte Gütefunktion stark von der optimalen Lösung abweichen, da komplett auf zukünftig zugängliche Information verzichtet wird. Folgende Gleichung zeigt einen *Open-Loop*-Updateschritt der Gütefunktion des transformierten *Belief-State-MDP*s.

$$J_k^i(\underline{b}_k^i) = \max_{\underline{u}_k^i} \left\{ R(\underline{b}_k^i, \underline{u}_k^i) + J_{k+1}^i(\phi(\underline{b}_k^i, \underline{u}_k^i)) \right\}$$

Die Funktion $\phi(\underline{b}_k^i, \underline{u}_k^i)$ bezeichnet hierbei die Prädiktion der Verteilung $\underline{b}_k^i$ durch Aktion $\underline{u}_k^i$. Der Filterschritt, der implizit in $\phi(\underline{b}_k^i, \underline{u}_k^i, \underline{z}_{k+1}^i)$ des Update-Schrittes der korrekten *POMDP*-Lösung (4.3) steckt, wird also nicht berechnet, sondern die Zustandsverteilungen nur prädiziert [YCG+05]. Im Gegensatz zu (4.3) muss somit nicht der Erwartungswert über alle potentiellen Beobachtungen gebildet werden. Dies hat zur Folge, dass der Suchbaum nur nach möglichen Aktionen, nicht nach potentiellen Messungen aufgespalten wird, was die Komplexität der Planung deutlich reduziert.

Ein Problem dieses Ansatzes ist, dass die Zustandsschätzung durch wiederholte Prädiktion ohne Filterschritte sehr ungenau werden kann. Nach einigen Schritten der Planung kann die Unsicherheit der Schätzung so groß werden, dass unter Umständen keine sinnvollen Entscheidungen mehr getroffen werden können.

Nominal Belief State Optimization

Um das Beobachtungsmodell in die Planung mit einzubeziehen, aber trotzdem die Komplexität gering zu halten, stellt Miller in [MHC08] einen Ansatz mit dem Namen *Nominal Belief State Optimization* vor. Im Gegensatz zu *OLF* wird der Filterschritt aber nicht komplett vernachlässigt, sondern es wird mit der erwarteten Beobachtung geplant:

$$J_k^i(\underline{b}_k) = \max_{\underline{u}_k^i} \big\{ R(\underline{b}_k^i, \underline{u}_k^i) + J_{k+1}^i(\phi(\underline{b}_k^i, \underline{u}_k^i, \hat{\underline{z}}_{k+1}^i)) \big\}.$$

Die erwartete Beobachtung $\hat{\underline{z}}_{k+1}^i$ entspricht dem Erwartungswert $\mathrm{E}\big\{p(\underline{\boldsymbol{z}}_{k+1}^i | \underline{b}_{k+1}^i)\big\}$, der aus der prädizierten Dichte $\underline{b}_{k+1}^{\mathrm{p},i} = \phi(\underline{b}_k^i, \underline{u}_k^i)$ sowie der Messabbildung $O(\underline{z}_{k+1}^i | \underline{b}_{k+1}^{\mathrm{p},i})$ bestimmt werden kann. Die ausgezeichnete Messung $\hat{\underline{z}}_{k+1}^i$ wird dann für die Filterung verwendet, um im Entscheidungsbaum weiter abzusteigen. Dabei bezeichnet $\phi(\underline{b}_k^i, \underline{u}_k^i, \hat{\underline{z}}_{k+1}^i)$ die Prädiktion der Dichte $\underline{b}_k^i$ mit Aktion $\underline{u}_k^i$ und die anschließende Filterung mit $\hat{\underline{z}}_{k+1}^i$. Vorteil dieses Verfahrens ist es, dass die Unsicherheit der Schätzung auch nach mehreren Prädiktionsschritten durch wiederholte Filterung klein gehalten wird. Da der Suchbaum nicht nach verschiedenen Messungen aufgespaltet wird, generiert auch *Nominal Belief State Optimization* eine feste Aktionenfolge.

Verwendung einer vorberechneten Gütefunktion

In Abschnitt 4.4 wurde gezeigt, wie die optimale Gütefunktion über dem kompletten *Belief-Space* vorberechnet werden kann. Ist die Gütefunktion J_1^i an jeder Stelle des *Belief-Space* bekannt, kann daraus für beliebige Zustandsverteilung schnell die nächste optimale Aktion berechnet werden.

Um nicht nur die erste auszuführende Aktion sondern eine Aktionenfolge über H Schritte zu generieren, können auch in diesem Fall die Verfahren eingesetzt werden, die in den vorherigen zwei Abschnitten vorgestellt wurden. Nachdem mit der berechneten Aktion prädiziert wurde, kann zum einen der Filterschritt vernachlässigt werden, zum anderen kann wie zuvor beschrieben eine bestimmte Messung für die Filterung ausgewählt werden. In beiden Fällen ist der nächste *Belief-Point* eindeutig bestimmt, von dem aus unter Verwendung der vorberechneten Gütefunktion wieder die optimale Aktion gewählt werden kann. Die wiederholte Ausführung dieses Planungsschrittes ergibt somit eine feste Abfolge von Aktionen.

Wird die Gütefunktion wie in Abschnitt 4.4.1 beschrieben vorausberechnet, so führt ein Agent seine optimale *Closed-Loop*-Strategie aus, solange innerhalb des Planungshorizontes keine Interaktion mit anderen Agenten stattfindet. Nach einer Planungsphase ist zwar eine Aktionenfolge der Länge H bekannt, allerdings wird nur die erste Aktion dieser Sequenz tatsächlich ausgeführt. Diese erste Aktion entspricht der optimalen *Closed-Loop*-Strategie. Alle weiteren Aktionen dienen nur der Planung und haben keinen direkten Einfluss auf die tatsächliche Ausführung, da nach jedem Schritt neu geplant wird.

5.4 Multi-Agenten-Regelung

Mit den bisherigen Betrachtungen kann nun das vollständige Verfahren für das Multi-Agenten-System beschrieben werden. Das Ergebnis der Planung ist eine gemeinsame Strategie, die sich unter den gegebenen Einschränkungen im Nash-Gleichgewicht befindet. Dieser Algorithmus wird von jedem Agenten ausgeführt und läuft identisch ab, solange den Agenten die gleiche initiale Verteilung bekannt ist. Im Folgenden wird daher der Planungsvorgang beschrieben, der von einem Agenten ausgeführt wird.

Zu Beginn der Planungsphase wird eine Sequenz von gemeinsamen Aktionen über dem Planungshorizont generiert. Diese initiale gemeinsame Strategie π muss allen Agenten bekannt sein um zu garantieren, dass die Berechnung tatsächlich identisch abläuft und alle Agenten zum gleichen Ergebnis kommen. Da keine Kommunikation zugelassen ist, kann dafür ein global bekanntes Protokoll eingesetzt werden. Auf das Beispiel der Pfadplanung angewandt könnte beispielsweise festgelegt werden, dass sich kein Roboter innerhalb des Planungshorizontes bewegt.

Die gemeinsame Strategie π kann im Allgemeinen verbessert werden. Dafür wird ein Agent i ausgewählt und die Pläne π^{-i} der restlichen Agenten $j \neq i$ festgehalten. Für Agent i wird nun die lokal optimale Lösung berechnet, wie im Abschnitt 5.3 beschrieben. Lokal optimal ist hier so zu verstehen, dass Agent i die beste Strategie unter der Annahme wählt, dass die anderen Agenten die Strategien π^{-i} befolgen. Da der ausgewählte Agent nur seine eigenen Übergangswahrscheinlichkeiten beeinflusst, sind auch die Verteilungen der Zustände $\underline{x}^{j\neq i}_{1,\ldots,H}$ unabhängig von den Entscheidungen des i-ten Agenten. Somit kann die Belohnungsfunktion $R_G(\underline{x}_k)$ als Funktion $R^i_G(\underline{x}^i_k)$ dargestellt werden. Die stochastischen Prozesse sind ansonsten komplett unabhängig, daher ergibt sich ein $POMDP^i$, wie in Definition 5.1 beschrieben. Für dieses lokale Problem kann durch eines der in Abschnitt 5.3.2 vorgestellten Verfahren eine optimale Abfolge von Aktionen $\tilde{\pi}^i$ gefunden werden.

Die Kombination der neuen lokalen Strategie $\tilde{\pi}^i$ und der festgehaltenen Strategien der restlichen Agenten π^{-i} definiert wiederum das gemeinsame Verhalten aller Agenten über den Planungshorizont. Für diese neue gemeinsame Strategie kann die Gütefunktion $J^i_{(\tilde{\pi}^i,\pi^{j\neq i})}(\underline{b}_0)$ des i-ten Agenten bestimmt werden. Diese Gütefunktion entspricht der erwarteten Summe der lokalen Belohnungsfunktion R^i_L sowie der globalen Belohnungsfunktion R^i_G über den Planungshorizont. Konnte J^i im Vergleich zur initialen Strategie verbessert werden, so wird für den weiteren Planungsverlauf die verbesserte Strategie $(\tilde{\pi}^i, \pi^{j\neq i})$ verwendet.

Nun wird ein weiterer Agent ausgewählt und der eben beschriebene Schritt ausgehend von der aktualisierten gemeinsamen Strategie wiederholt. Dies wird so lange fortgeführt, bis kein Agent mehr seine lokale Strategie verbessern kann, d.h. alle Agenten die *optimale Antwort* auf das Verhalten der anderen Agenten gefunden haben. Nach Definition befindet sich die somit gefundene gemeinsame Strategie unter den gegebenen Einschränkungen in einem Nash-Gleichgewicht.

Algorithm 1 ALTERNATING_OPTIMIZATION Input: $\underline{u}^{1,\dots,N}_{0,\dots,H-1}, \underline{b}_0$ Output: $\underline{u}^i_0$

```
while not converged do
  set converged
  for i = 1, ..., n do
    π^{i*} ← GET_BEST_RESPONSE u^{-i}_{0,...,H-1}
    u^i_{0,...,H-1} ← FIXED_ACTION_SEQUENCE(π^{i*})
    if J^i(u^{1,...,N}_{0,...,H-1}) is increased then
      set not converged
    end if
  end for
end while
```

Algorithmus 1 zeigt den allgemeinen Ablauf der alternierenden Optimierung. Die Eingabe ist eine initiale Abfolge von Aktionen mit Horizont H für alle Agenten. Ausgabe ist die Aktion, die für Agent i als nächstes auszuführen ist. Die äußere Schleife wird ausgeführt, bis der Algorithmus konvergiert, d.h. bis in der inneren Schleife keine Verbesserung einer lokalen Strategie mehr auftritt.

Innerhalb der inneren Schleife wird nacheinander für alle Agenten jeweils eine optimale Stellgrößenabfolge bestimmt. Dazu wird mit GET_BEST_RESPONSE zunächst die *optimale Antwort* auf die fixierten Strategien $\underline{u}^{-i}_{0,\dots,H-1}$ berechnet. Das Ergebnis dieser Berechnung π^{i*} ist die optimale *POMDP*-Strategie für Agent i unter der Voraussetzung, dass alle anderen Agenten die festgelegte Stellgrößenabfolge ausführen. In Abschnitt 5.7 wird ein effizienter Algorithmus für diese Berechnung vorgestellt. Aus dieser Strategie wird gegebenenfalls eine feste Aktionenfolge mittels FIXED_ACTION_SEQUENCE bestimmt. Alternativ können die zwei Schritte auch zusammengefasst werden und die Abfolge von Aktionen direkt mit einem der in Abschnitt 5.3.2 vorgestellten Approximationsverfahren generiert werden.

Nun wird die erwartete Güte des i-ten Agenten J^i berechnet, die aus der aktualisierten gemeinsamen Strategie folgt. Konnte die Gütefunktion erhöht werden, so wird die Variable *converged* zurückgesetzt, um eine weitere Iteration zu gewährleisten.

5.5 Konvergenz

In folgendem Satz wird gezeigt, dass das Verfahren konvergiert. Für jeden Agenten wird zwar jeweils nur die eigene Gütefunktion J^i maximiert, was im Allgemeinen auch dazu führen kann, dass die Güte J^{-i} anderer Agenten abnimmt. Betrachtet man allerdings die Summe aller vorkommenden Teilfunktionen J^i_L und J_G, so steigt der Wert dieser Summe stetig mit jeder Iteration. Da die Güte durch die global bestimmte optimale Strategie beschränkt ist, konvergiert das Verfahren somit.

Satz 5.1 *Sei ein transitions- und beobachtungsunabhängiges System mit additiven Belohnungsfunktionen gegeben (s. Abschnitt 5.1). Wird die eben beschriebene alternierende Optimierung durchgeführt, so konvergiert das Verfahren.*

Beweis.

Aufgrund der Transitionsunabhängigkeit beeinflusst π^i nur die Verteilung der Zufallsvariablen $\underline{\boldsymbol{x}}_k^i$. Die Zustände der anderen Agenten können nicht beeinflusst werden, da deren Strategie in der Planung festgehalten wird. Somit verändert sich mit Änderung von π^i nur R_L^i und R_G, nicht R_L^j für $j \neq i$. In jedem Schritt optimiert nun Agent i seine Strategie, d.h. er maximiert

$$J^i = J_L^i + J_G^i = \mathrm{E}\left\{\sum_{k=0}^{H-1}(R_L^i(\underline{\boldsymbol{x}}_k^i, \underline{u}_k)\right\} + \mathrm{E}\left\{\sum_{k=0}^{H-1}(R_G(\underline{\boldsymbol{x}}_k, \underline{u}_k))\right\}.$$

Somit gilt:

1. $\tilde{J}_L^j = J_L^j$ für $j \neq i$
2. $\tilde{J}_L^i + \tilde{J}_G \geq J_L^i + J_G$,

wobei $\tilde{J}^i, \tilde{J}_G^i$ und $\tilde{J}_L^i$ die jeweiligen Gütefunktionen nach Ausführung der Optimierung bezeichnen. Betrachtet man nun die Summe aller lokalen Gütefunktionen und der globalen Gütefunktion, so gilt:

$$\overset{1,2}{\Longrightarrow} (\sum_{j\neq i} \tilde{J}_L^j) + \tilde{J}_L^i + \tilde{J}_G \geq (\sum_{j\neq i} J_L^j) + J_L^i + J_G.$$

Somit wächst die Summe aller einzelnen Gütefunktionen monoton mit jeder Iteration. Zusätzlich ist der Wert durch das globale Optimum dieser Summe von oben beschränkt. Daraus folgt, dass das Verfahren konvergiert. □

5.6 Komplexität

Stehen den Agenten in jedem Planungsschritt nur eine endliche Anzahl von Aktionen zur Verfügung, so gibt es auch nur endlich viele Kombinationen von Aktionenfolgen der Länge H. Damit ist garantiert, dass das Verfahren nach endlich vielen Schritten terminiert. Im schlechtesten Fall müssen aber auch tatsächlich all diese Kombinationen berechnet werden, bis ein lokales Optimum erreicht ist. Wird vorausgesetzt, dass ein lokaler Planungsschritt in $\mathcal{O}(H)$ stattfindet, liegt die Komplexität des Verfahrens somit in

$$\mathcal{O}(H \cdot |\mathcal{U}|^{H \cdot N}) ,$$

ist also exponentiell abhängig von der Horizontlänge sowie der Anzahl der Agenten. Die Laufzeittests in Abschnitt 6.4.1 zeigen aber, dass meist bedeutend weniger Iterationen ausreichen, um eine Lösung zu finden.

5.7 Effiziente Berechnung

Das bisher beschriebene Verfahren basiert auf den Lösungen der lokalen $POMDP^i$, die in jedem Iterationsschritt zu berechnen sind. In Kapitel 4 wurde gezeigt, wie die optimale Strategie eines *POMDP* durch Transformation in einen gleichwertigen *Belief-Space-MDP* gefunden werden kann. Im Folgenden wird beschrieben, wie diese Transformation auch über kontinuierlichem Zustandsraum effizient durchgeführt werden kann. Dabei wird ausgenutzt, dass die meisten aufwändigen Rechenoperationen vorberechnet werden können, um online die optimale Strategie je nach Verhalten der anderen Agenten schnell anzupassen.

5.7.1 Darstellung der Gütefunktion bei kontinuierlichem Zustandsraum

Bei der Transformation eines *POMDP* über kontinuierlichem Zustandsraum in ein *Belief-State-MDP* treten mehrere Probleme auf. Im Gegensatz zu einem diskreten Zustandsraum mit K Zuständen, bei dem sich $\Delta\mathcal{X}$ als Simplex mit Dimension $K-1$ darstellen lässt [KLC98], ist der Raum der Zustandsverteilungen bei kontinuierlichem Zustandsraum unendlichdimensional [ZFS08]. Um die Gütefunktion über diesem Raum darzustellen, sind aber im Allgemeinen auch unendlich viele Parameter notwendig, was die Berechnung in einem digitalen Rechnersystem nicht zulässt. Oft kann aber die interne Struktur des Systems ausgenutzt werden, um die auftretenden Dichten geschickt zu parametrisieren. Zum Beispiel treten bei linearen Systemen mit Gaußschem Rauschen nur Normalverteilungen auf, die eindeutig durch Mittelwert und Kovarianz charakterisiert sind [HR09, MHC08, BMDW06]. Weitere Möglichkeiten in allgemeineren Systemen werden beispielsweise in [Pou02] und [ZFS08] beschrieben.

Während die Gütefunktion des zugehörigen *Belief-State MDP* bei diskreten Systemen stückweise linear ist, ist dies bei einem parametrisierten *Belief-Space* nicht der Fall. Punktbasierte Verfahren lösen dieses Problem approximativ, indem der kontinuierliche Raum aller Verteilungen diskretisiert wird und die Gütefunktion nur auf diesen diskreten Punkten berechnet wird [BMDW06]. Dieses Verfahren ist als *Fitted Value Iteration* bekannt und wird im folgenden Abschnitt beschrieben.

Fitted Value Iteration

Im Allgemeinen sind die Zustandsräume der $POMDP^i$ für jeden Agenten i verschieden, daher müssen folgende Berechnungen für jeden einzelnen Agenten durchgeführt werden. Zur besseren Lesbarkeit werden in diesem Abschitt allerdings die Indizes i vernachlässigt, da die Betrachtungen für alle Agenten dem gleichen Prinzip folgen.

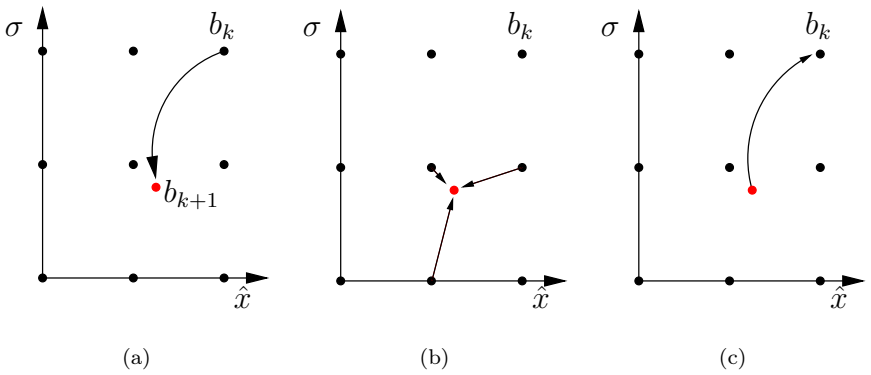


Abbildung 5.2: Hier ist das Verfahren *Fitted Value Iteration* am Beispiel eines zweidimensionalen Raumes dargestellt, in dem Dichten über Mittelwert $\hat{x}$ und Standardabweichung σ parametrisiert sind. (a) Für einen Update-Schritt ist es notwendig, die Gütefunktion J_{k+1} im Punkt $\underline{b}_{k+1}$ auszuwerten. (b) Da die Funktion aber nur an den Gitterpunkten definiert ist, muss die Funktion in $\underline{b}_{k+1}$ über umliegende Punkte interpoliert werden. (c) Dieser Wert wird schließlich verwendet, um die Güte in $\underline{b}_k$ zu bestimmen.

Sei $\tilde{\mathcal{B}} = \{\hat{\underline{b}}_1, \ldots, \hat{\underline{b}}_{|\tilde{\mathcal{B}}|}\}$ eine endliche Teilmenge des Raumes $\mathcal{B}$, auf der ein Update-Schritt der Value-Iteration

$$J_k(\hat{\underline{b}}_l) = \max_{\underline{u}_k} E\big\{R(\hat{\underline{b}}_l, \underline{u}_k) + J_{k+1}(\phi(\hat{\underline{b}}_l, \underline{u}_k, \underline{z}_{k+1})\big\}$$

ausgeführt werden soll. Die prädizierte und gefilterte Dichte $\phi(\hat{\underline{b}}_l, \underline{u}_k, \underline{z}_{k+1})$, an deren Stelle die Gütefunktion J_{k+1} rekursiv ausgewertet wird, liegt aber im Allgemeinen nicht mehr in der Menge $\tilde{\mathcal{B}}$. Die Gütefunktion J_{k+1} wird deshalb über Punkte „in der Nähe“ von $\phi(\hat{\underline{b}}_l, \underline{u}_k, \underline{z}_{k+1})$ interpoliert. Dafür wird eine Interpolationsfunktion eingesetzt, welche die auf den Punkten $\hat{\underline{b}}_l$ definierte Gütefunktion J_{k+1} verwendet, um $\hat{\underline{b}}_l^{\mathrm{e}}$, der prädizierten und gefilterten Verteilung, einen Wert zuzuweisen. Abb. 5.2 zeigt diesen Vorgang exemplarisch anhand eines zweidimensionalen *Belief-Space*.

5.7.2 Vorberechnung der Interpolationsindizes

Wird ein lineares Interpolationsverfahren verwendet, so ist die Gütefunktion J_k an der Stelle einer beliebigen Dichte $\underline{b}$ als Linearkombination

$$J_k(\underline{b}) = \sum_l c_l(\underline{b}) \cdot J_k(\hat{\underline{b}}_l)$$

definiert. Die Koeffizienten c_l sind nur von der Dichte $\underline{b}$ abhängig, nicht von der Gütefunktion J_k. Wird also in jedem Planungsschritt die feste Menge $\tilde{\mathcal{B}}$ für die *Value Iteration* verwendet, so können die Interpolationsindizes

$$c_l(\phi(\hat{\underline{b}}_m, \underline{u}, \underline{z})) = c_l^{m,u,z}$$

für alle Punkte, Aktionen und mögliche Beobachtungen vorberechnet werden. Der Update-Schritt kann somit fogendermaßen dargestellt werden:

$$\begin{aligned} J_k(\underline{b}_m) &= \max_{\underline{u}_k} E\big[R(\underline{b}_m, \underline{u}_k) + \sum_l c_l^{m,u_k,z_{k+1}} \cdot J_{k+1}(\underline{b}_l)\big] \\ &= \max_{\underline{u}_k} \sum_{\mathcal{Z}} \big[(R(\hat{\underline{b}}_l, \underline{u}_k) + \sum_l c_l^{m,u_k,z_{k+1}} \cdot J_{k+1}(\hat{\underline{b}}_l)) \cdot p(\underline{z}_{k+1}|\hat{\underline{b}}_m, \underline{u}_k)\big]. \end{aligned}$$

Ist auch das Messmodell unabhängig vom Planungshorizont, so können die Beobachtungswahrscheinlichkeiten $p(\underline{z}|\underline{b}_{m,k}, \underline{u}) = d^{m,u,z}$ ebenfalls vorberechnet werden, wie folgt:

$$\begin{aligned} J_k(\hat{\underline{b}}_m) &= \max_{\underline{u}_k} \sum_{\mathcal{Z}} \big[(R(\underline{b}_m, \underline{u}_k) + \sum_l c_l^{m,u_k,z_{k+1}} \cdot J_{k+1}(\underline{b}_l)) \cdot d^{m,u_k,z_{k+1}}\big] \\ &= \max_{\underline{u}_k} \big[R(\underline{b}_m, \underline{u}_k) + \sum_{\mathcal{Z}} (\sum_l c_l^{m,u_k,z_{k+1}} \cdot J_{k+1}(\underline{b}_l)) \cdot d^{m,u_k,z_{k+1}}\big] \end{aligned}$$

Nun können die Koeffizienten $d^{m,u,z}$ in die Summe über l hineingezogen werden und schließlich die beiden Summen vertauscht werden. So erhält man folgenden Term, in dem die Summe über die Koeffizienten aller möglichen Messungen zusammengefasst werden können:

$$\begin{aligned} J_k(\hat{\underline{b}}_m) &= \max_{\underline{u}_k} R(\underline{b}_m, \underline{u}_k) + \sum_l \big[(\sum_{\mathcal{Z}} c_l^{m,u_k,z_{k+1}} \cdot d^{m,u_k,z_{k+1}} \cdot J_{k+1}(\underline{b}_l))\big] \\ &= \max_{\underline{u}_k} \big[R(\underline{b}_m, \underline{u}_k) + \sum_l e_l^{m,u_k} \cdot J_{k+1}(\underline{b}_l))\big]. \end{aligned} \tag{5.3}$$

5.7.3 Vorberechnung der lokalen Belohnungsfunktion

Die lokale Belohnungsfunktion R_L^i bleibt mit Änderung der Strategien anderer Agenten konstant. Daher kann auch R_L^i vorberechnet werden. Dabei muss die Belohnungsfunktion nur an den Stellen $R_L^i(\hat{\underline{b}}_l)$ für $l = 1, \ldots, |\tilde{\mathcal{B}}|$ betrachtet werden, da nur diese während der *Value-Iteration* benutzt werden. Die lokale Belohnungsfunktion R_L^i kann somit als $|\tilde{\mathcal{B}}|$-dimensionaler Vektor $\underline{r}_L$ dargestellt werden.

5.7.4 Updateschritt der Fitted-Value-Iteration

Die globale Belohnungsfunktion R_G kann ebenfalls in Vektorschreibweise dargestellt werden, verändert sich allerdings mit den Zustandsverteilungen der anderen Agenten. Da diese Schätzungen sich mit jedem Schritt im Planungshorizont ändern, sind insgesamt H Vektoren $\underline{r}_{G,k}$

notwendig, um R_G^i zu beschreiben. Ein Update-Schritt der *Value-Iteration* kann somit durch Vektor-Matrix-Operationen durchgeführt werden. Dafür werden die Koeffizienten $e_l^{m,u}$ in eine Matrix $\mathbf{A}^u$ geschrieben, die durch $\mathbf{A}^u(m,l) = e_l^{m,u}$ definiert ist. Insgesamt bestehen somit $|\mathcal{U}|$ dieser Matrizen, die jeweils die Koeffizienten der Maximierung in (5.3) beinhalten. Somit können die Initialisierung sowie der Update-Schritt (5.3) für alle $\underline{\hat{b}}_l$ als Vektor-Matrix-Operationen dargestellt werden:

$$\begin{aligned} \underline{j}_H &= \underline{r}_L + \underline{r}_{G,H} \\ \underline{j}_k &= (\underline{r}_L + \underline{r}_{G,k}) + \max_{\underline{u}_k} \mathbf{A}^{\underline{u}_k} \cdot \underline{j}_{k+1}. \end{aligned} \tag{5.4}$$

Der Vektor $\underline{j}_k$ beschreibt dabei die Gütefunktion J_k an den Stellen $\underline{\hat{b}}_l$. Der max-Operator ist in diesem Fall elementweise zu verstehen, er muss also für jeden Eintrag von $\underline{j}_k$ einzeln ausgewertet werden. Die Komplexität eines Update-Schrittes liegt somit in $\mathcal{O}(|\mathcal{U}||\mathcal{B}|^2)$, ist aber unabhängig vom Horizont. Da insgesamt H Update-Schritte durchgeführt werden, um die Gütefunktion zu berechnen, ist der Gesamtaufwand der *Value-Iteration* daher linear im Horizont.

Algorithmus 2 zeigt einen Schritt der alternierenden Optimierung durch *Fitted-Value-Iteration*. Der Algorithmus verlangt als Eingabe die festen Aktionenfolgen aller Agenten außer i, sowie eine gemeinsame initiale Zustandsverteilung. Zunächst werden die Zustandsverteilungen $\underline{b}_h^{j\neq i}$ über den Planungshorizont für alle Agenten außer i durch PREDICT bestimmt. Die globale Belohnungsfunktion R_G wird nun an der Stelle dieser *Belief-States* für alle Agenten $j \neq i$, sowie an den diskreten Dichten $\underline{\hat{b}}^i$ ausgewertet. Somit sind die Einträge der Vektoren $\underline{r}_G$ bestimmt, welche die globale Belohnungsfunktion für Agent i repräsentieren. Die Gütefunktion, welche durch die Vektoren $\underline{j}_k^i$ gegeben ist, ändert sich mit der globalen Belohnungsfunktion. Daher wird die Gütefunktion durch UPDATE_VALUE_FUNCTION angepasst. Dieser Schritt entspricht (5.4). Schließlich kann durch BEST_ACTION die beste Aktion von jedem *Belief-State* bestimmt werden. Die Details dieses Schrittes sind in Abschnitt 4.4.2 ausführlich erklärt. So kann eine Aktionenfolge $\underline{u}^i_{0,\dots,H-1}$ generiert werden, die als Ergebnis ausgegeben wird.

Oft findet die Interaktion der Agenten nur in Teilen des Zustandsraumes oder nur in manchen Schritten des Planungshorizontes statt. Zum Beispiel interagieren die Roboter bei der Pfadplanung nur, wenn sie sich nahe genug für eine potentielle Kollision kommen. Diese Lokalität der Interaktion kann dazu führen, dass die globale Gütefunktion nur an manchen Stellen Werte ungleich Null annimmt. Als Folge ändert sich auch die Gütefunktion eines Agenten nicht für jede Dichte der Menge $\tilde{\mathcal{B}}^i$. Unter bestimmten Voraussetzungen können *Belief-States* bestimmt werden, deren Funktionswert trotz veränderter globaler Gütefunktion konstant bleibt. Diese Dichten müssen in dem Update-Schritt der Gütefunktion nicht berücksichtigt werden, was zusätzlichen Geschwindigkeitsvorteil während der Planung bedeuten kann.

Ändert sich insbesondere die gemeinsame Belohnungsfunktion nur im Horizont $\tilde{H} < H$, können sich auch die Gütefunktionen J^i nur in diesem Horizont ändern. Somit können unterschiedliche Planungshorizonte für die lokale und globale Planung eingesetzt werden. Im einführenden

Algorithm 2 GET_BEST_RESPONSE Input: $\underline{b}_0, \underline{u}^{j\neq i}_{0,\ldots,H-1}$ Output: $\underline{u}^i_{0,\ldots,H-1}$

for $a = 1, \ldots, i-1, i+1, \ldots N$ **do**
  $\underline{b}^a_{1,\ldots,H} \leftarrow$PREDICT$(\underline{b}^a_0, \underline{u}^a_{0,\ldots,H-1})$
end for
for $h = 0, \ldots, H-1$ **do**
  for $m = 1, \ldots |\tilde{\mathcal{B}}^i|$ **do**
    $\underline{r}^i_{G,h}(\hat{\underline{b}}^i_m) = R_G(\underline{b}^1_h, \ldots, \underline{b}^{i-1}_h, \hat{\underline{b}}^i_m, \underline{b}^{i+1}_h, \ldots, \underline{b}^N_h)$
  end for
end for
for $h = H-1, \ldots, 0$ **do**
  $\underline{j}^i_h \leftarrow$ UPDATE_VALUE_FUNCTION $(\underline{r}^i_L, \underline{r}^i_G, \underline{j}^i_{h+1})$
end for
for $h = 0, \ldots, H-1$ **do**
  $\underline{u}^i_h \leftarrow$ BEST_ACTION$(\underline{j}^i, \underline{b}_h)$
end for

Beispiel 1.2 der Roboter-Pfadplanung kann die lokale Gütefunktion über einen langen Horizont vorberechnet werden, so dass die hohe Zielbelohnung auf jeden Fall berücksichtigt wird. In der Ausführung kann je nach Rechenleistung die Interaktion zwischen den Agenten auf kurzen Horizont beschränkt werden, um schnell gemeinsame Strategien zu finden. Die Details dieser Betrachtungen sind im Anhang A.3 zu finden.

5.8 Anwendung

Das in diesem Kapitel beschriebene Verfahren setzt ein transitions- und beobachtungsunabhängiges Modell voraus. Damit kann garantiert werden, dass die resultierende Strategie ein Nash-Gleichgewicht aus Stellgrößenabfolgen ist und alle Agenten die gleiche gemeinsame Strategie finden. Die Voraussetzungen schränken aber auch stark die Prozesse ein, die mit einem solchen Modell beschrieben werden können. Unter bestimmten Voraussetzungen ist es aber möglich, das Verfahren auch auf allgemeinere Systeme anzuwenden. Im Allgemeinen gelten dann bestimmte theoretische Ergebnisse nicht mehr und das Verfahren ist als Approximation zu verstehen.

Wie schon zuvor erwähnt, kann das Verfahren auch auf Systeme angewandt werden, bei denen die Agenten auch Beobachtungen über andere Agenten erhalten können. In diesem Fall gibt es aber weitere Auswirkungen zu beachten. Um zu garantieren, dass alle Agenten zur gleichen Lösung kommen, wurde vorausgesetzt, dass alle die gleiche a-priori Verteilung besitzen. Dass diese Voraussetzung nicht verletzt wird, dürfen für die Zustandsschätzung nur die Messungen verwendet werden, die tatsächlich allen Agenten in gleicher Form vorliegen.

Ist dies aber nicht möglich, weil beispielsweise jeder Agent eigene, rauschbehaftete Sensoren besitzt, so liegen nach einem Filterschritt unterschiedliche a-priori-Verteilungen als Basis für

die nächste Planungsphase vor. Das vorgestellte Verfahren kann trotzdem eingesetzt werden, indem jeder Agent davon ausgeht, dass die anderen Agenten ebenfalls das gleiche Wissen besitzen und sich dementsprechend verhalten. Da diese Annahme allerdings falsch sein kann, ist nicht mehr garantiert, dass alle Agenten die gleiche gemeinsame Strategie finden. Weichen die Informationen allerdings nicht stark voneinander ab, so können in der Praxis trotzdem gute Ergebnisse erwartet werden.

Desweiteren steht und fällt das Verfahren mit der Darstellung der vorkommenden Verteilungen. Die Menge der diskreten Dichten, die für die *Value-Iteration* benötigt werden, steigt exponentiell mit der Dimension des parametrisierten *Belief-Space*. Deshalb ist es wichtig, eine niedrigdimensionale Darstellung zu finden, die alle relevanten Informationen der Verteilungen beinhaltet. Ein weiterer ausschlaggebender Punkt ist die Komplexität der Auswertung der globalen Belohnungsfunktion, die in jedem Schritt neu berechnet werden muss. Idealerweise lässt sich für diese Funktion eine geschlossene Form finden, die direkt von den Parametern der *Belief-States* $\underline{b}^i$ abhängt. Ansonsten muss für jede Dichte das Integral entsprechend (5.2) numerisch gelöst werden, was unter Umständen sehr rechenaufwendig ist. Ein Beispiel, für das diese Voraussetzungen eintreffen, ist die Pfadplanung in teilweise beobachtbaren Umgebungen, auf die das Verfahren angewandt wurde und die im folgenden Kapitel beschrieben wird.

KAPITEL 6

Experimente

Im Folgenden wird die Effektivität der in Kapitel 5 vorgestellten alternierenden Optimierung anhand von Problemen aus der Roboter-Pfadplanung gezeigt und mit einem einfachen Referenzverfahren verglichen. Bei den betrachteten Szenarien soll ein Team von Robotern zu einer ausgezeichneten Zielfläche finden, wobei Hindernisse den direkten Weg blockieren. Die Roboter müssen auch ihre gegenseitige Lage berücksichtigen, um kollisionsfrei zwischen den Hindernissen zum Ziel zu finden. Um die jeweiligen Positionen zu schätzen, sind Landmarken in der Umgebung angebracht, zu denen Abstandsmessungen durchgeführt werden können. In jedem Zeitschritt bestimmt der verwendete Regler jedes Agenten die optimale nächste Stellgröße, die dann von den Robotern ausgeführt wird, um im nächsten Schritt neu zu planen.

6.1 Testumgebung

6.1.1 System- und Messmodell

Der kontinuierliche Zustand jedes Roboters besteht aus einem dreidimensionalen Vektor $\underline{x} = [x, y, \phi]^T$. Dabei entsprechen die ersten beiden Einträge der Position, der Winkel ϕ gibt die aktuelle Ausrichtung an. In jedem Schritt beeinflussen die Stellgrößen $[u^s, u^\phi]^T$ den aktuellen Zustand entsprechend dem Modell:

$$\underline{x}_{k+1} = f \begin{pmatrix} x_k \\ y_k \\ \phi_k \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \\ \phi_k \end{pmatrix} + \begin{pmatrix} (u_k^s + \boldsymbol{v}_k^s) cos(\phi_k) \\ (u_k^s + \boldsymbol{v}_k^s) sin(\phi_k) \\ u_k^\phi + \boldsymbol{v}_k^\phi \end{pmatrix} \tag{6.1}$$

Die Bewegungsgleichung ist an Miniatur-Laufroboter angelehnt, die in jedem Schritt Lenkbewegungen auf die Vorwärtsbewegung superponieren können [WMH07]. Die Eingangsgrößen, also Schrittlänge u_k^s und Drehwinkel u_k^ϕ, unterliegen jeweils erwartungswertfreiem, normalverteiltem Rauschen $\boldsymbol{v}^s$ bzw. $\boldsymbol{v}_k^\phi$, das im Rauschvektor $\underline{\boldsymbol{v}}_k$ zusammengefasst ist. Die Kovarianzmatrix $\mathrm{Cov}(\underline{\boldsymbol{v}}_k)$ wird mit $\mathbf{Q}$ bezeichnet. In jedem Schritt können die Roboter entweder stehen bleiben oder eine Vorwärtsbewegung fester Länge ausführen, also $u_k^s \in \{0, 1\}$. Zusätzlich kann in beiden Fällen eine Drehung um konstanten Winkel $u_k^\phi \in \{-\pi/8 \text{ rad}, 0 \text{ rad}, \pi/8 \text{ rad}\}$ durchgeführt

werden. Insgesamt ergeben sich damit sechs Kombinationen, von denen in jedem Schritt die geeignetste ausgewählt werden soll.

Um ihre jeweilige Position zu schätzen, stehen den Robotern Beobachtungen zur Verfügung, die aus verrauschten Distanzmessungen zu zwei Landmarken bestehen. Da die Landmarken so angebracht sind, dass die Position innerhalb des erlaubten Gebietes durch eine solche Messung eindeutig bestimmt ist, verbessert jede Beobachtung die aktuelle Zustandsschätzung. Dies erlaubt allerdings noch keine Rückschlüsse auf die Ausrichtung. Daher wird zusätzlich davon ausgegangen, dass jeder Roboter über einen Kompass verfügt, dessen Wert direkt ausgelesen werden kann. Das nichtlineare Messmodell sieht somit folgendermaßen aus:

$$\underline{z}_k = h(\underline{x}_k) = h \begin{pmatrix} \sqrt{(x_k - l_1^x)^2 + (y_k - l_1^y)^2} \\ \sqrt{(x_k - l_2^x)^2 + (y_k - l_2^y)^2} \\ \phi_k \end{pmatrix} + \underline{\boldsymbol{w}}_k.$$

Hierbei ist $l_i = [l_i^x, l_i^y]^T$ die Position der i-ten Landmarke. Der Zufallsvektor $\underline{\boldsymbol{w}}_k = [\boldsymbol{w}_k^{d1}, \boldsymbol{w}_k^{d2}, \boldsymbol{w}_k^{\phi}]^T$ bezeichnet normalverteiltes, unabhängiges Rauschen auf jeden Eintrag des Beobachtungsvektors mit Kovarianzmatrix $\mathbf{R}$.

6.1.2 Zustandsschätzung

Um den Zustand dynamisch zu schätzen, wird ein *Extended-Kalmanfilter* (*EKF*) eingesetzt [Sim06]. Die Zustandsschätzung mittels *EKF* kann in zwei Schritte unterteilt werden. Zunächst wird nach jedem Schritt die aktuelle Zustandsschätzung unter Verwendung der linearisierten Systemabbildung prädiziert, um eine Schätzung $\underline{\boldsymbol{x}}_k^{\mathrm{p}}$ des Zustands nach der ausgeführten Aktion zu erhalten. Mit gegebener Messung kann diese Schätzung verbessert werden, wobei der erwartete quadratische Fehler minimiert wird. Dafür wird die ebenfalls linearisierte Messabbildung verwendet. Das Ergebnis dieses Filterschrittes ist die Zustandsschätzung $\underline{\boldsymbol{x}}_k^{\mathrm{e}}$. Durch den Einsatz des linearisierten Modells können die auftretenden Dichten mit Mittelwert und Kovarianz charakterisiert werden. Im folgenden Abschnitt werden die zwei Schritte ausführlich beschrieben.

Die Systemabbildung f wird linearisiert, indem eine Taylorreihenentwicklung von (6.1) um den Entwicklungspunkt $\underline{x}_k = \hat{\underline{x}}_k$ und $\underline{v}_k = \hat{\underline{v}}_k = \underline{0}$ durchgeführt wird. Durch Abbruch der Reihe nach der ersten Stufe resultiert daraus die lineare Approximation

$$f(\underline{x}_k) \approx f(\hat{\underline{x}}_k, \hat{\underline{v}}_k) + \mathbf{A}(\underline{x}_k - \hat{\underline{x}}_k) + \mathbf{B}\underline{v}_k.$$

$\mathbf{A}$ und $\mathbf{B}$ sind dabei die Jakobimatrizen

$$\mathbf{A} = \left.\frac{\partial f}{\partial \underline{x}_k}\right|_{\underline{x}_k=\hat{\underline{x}}_k,\underline{v}=\underline{0}} = \begin{pmatrix} 1 & 0 & -vsin(\hat{\phi}_k) \\ 0 & 1 & vcos(\hat{\phi}_k) \\ 0 & 0 & 1 \end{pmatrix}$$

und

$$\mathbf{B} = \left.\frac{\partial f}{\partial \underline{v}_k}\right|_{\underline{x}_k=\hat{\underline{x}}_k,\underline{v}_k=\underline{0}} = \begin{pmatrix} cos(\hat{\phi}_k) & 0 \\ sin(\hat{\phi}_k) & 0 \\ 0 & 1 \end{pmatrix}$$

Für die Prädiktion des Mittelwertes wird weiterhin die nichtlineare Systemabbildung f verwendet, welche mit der linearisierten Funktion zusammenfällt, da um den Punkt $\hat{\underline{x}}_k$, also um den aktuellen Mittelwert, entwickelt wird.

$$\hat{\underline{x}}^{\mathrm{p}}_{k+1} = f(\hat{\underline{x}}^{\mathrm{e}}_k)$$

Um die Kovarianz der Schätzung durch das Systemmodell abzubilden, muss allerdings die linearisierte Funktion eingesetzt werden, um die Darstellung der Zufallsvariablen durch Normalverteilungen zu erhalten.

$$\mathbf{C}^{\mathrm{p}}_{k+1} = \mathbf{A}\mathbf{C}^{\mathrm{p}}_k\mathbf{A}^T + \mathbf{B}\mathbf{Q}\mathbf{B}^T$$

Für das Messupdate muss in ähnlicher Weise das Messmodell (6.2) linearisiert werden:

$$h(\underline{x}_k) \approx h(\hat{\underline{x}}_k, \hat{\underline{w}}_k) + \mathbf{H}_k(\underline{x}_k - \hat{\underline{x}}_k) + \underline{w}_k,$$

wobei H die Jakobimatrix der Messabbildung bezeichnet.

$$\mathbf{H}_k = \left.\frac{\partial f}{\partial \underline{x}_k}\right|_{\underline{x}_k=\hat{\underline{x}}_k,\underline{w}_k=\underline{0}} = \begin{pmatrix} |\frac{x_k-l_1^x}{|x_k-l_1|} & \frac{y_k-l_1^y}{|x_k-l_1|} & 0 \\ \frac{x_k-l_2^x}{|x_k-l_2|} & \frac{y_k-l_2^y}{|x_k-l_2|} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Zunächst wird die Abweichung der tatsächlichen Messung zu dem Messwert bestimmt, der mit der aktuellen Zustandsschätzung erwartet wird. Dafür kann die nichtlineare Messabbildung h eingesetzt werden.

$$\underline{r}_k = (\underline{z}_k - h(\hat{\underline{x}}^{\mathrm{p}}_k))$$

Mit den Hilfsgrößen *Residualkovarianz* $\mathbf{S}_k$ und dem *Kalman-Gain* $\mathbf{K}_k$

$$\begin{aligned} \mathbf{S}_k &= \mathbf{H}_k \cdot \mathbf{C}_k^{\mathrm{p}} \cdot \mathbf{H}_k^T + \mathbf{R} \\ \mathbf{K}_k &= \mathbf{C}_k^{\mathrm{p}} \cdot \mathbf{H}^T \cdot \mathbf{S}^{-1} \end{aligned}$$

kann schließlich der angepasste Mittelwert sowie die neue Kovarianz der Schätzung bestimmt werden.

$$\begin{aligned} \underline{\hat{x}}_k^{\mathrm{e}} &= \underline{\hat{x}}_k^{\mathrm{p}} + \mathbf{K}_k \underline{r}_k \\ \mathbf{C}_k^{\mathrm{e}} &= (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{H}_k) \cdot \mathbf{C}_k^{\mathrm{p}} \end{aligned}$$

Diese Schätzung minimiert den erwarteten quadratischen Fehler und wird mit jedem Zeitschritt wiederholt.

6.1.3 Belohnungsfunktion

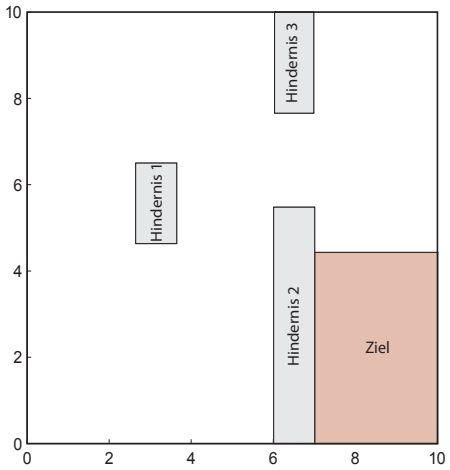


Abbildung 6.1: Die Abbildung zeigt die Position der Hindernisse sowie der Zielregion

Die Testszenarien bestehen aus statischen Umgebungen, in denen die Roboter eine ausgezeichnete Zielfläche erreichen sollen, ohne mit Hindernissen zusammenzustoßen. Die lokale Belohnungs- bzw. Bestrafungsfunktion R_L ist wie folgt definiert:

$$R_L^i(\underline{x}^i) = R_L^i \begin{pmatrix} x^i \\ y^i \\ \phi^i \end{pmatrix} = \begin{cases} 0 & , [x^i, y^i]^T \text{ liegt im Ziel} \\ -40 & , [x^i, y^i]^T \text{ liegt auf Hindernis} \\ -1 & , \text{sonst.} \end{cases}$$

Es ist zu sehen, dass nur den Positionen im Zielbereich keine Bestrafung zugeordnet ist. Jeder Punkt, der einer Kollision mit den statischen Hindernissen entspricht, wird mit -40 bestraft, jede sonstige Position erhält einen kleinen negativen Wert von -1. Da auch der maximale Planungshorizont auf 40 festgelegt ist, bedeutet das, dass ein kollisionsfreier Pfad in jedem Fall einer sicheren Kollision vorgezogen wird. Somit wird vermieden, dass Hindernisse „überlaufen“ werden, um schneller zum Ziel zu gelangen. Insgesamt kann also maximal die Güte 0 erreicht werden, wenn schon die Startposition im Ziel liegt. Jeder zusätzliche Schritt und insbesondere Kollisionen mit Hindernissen erhöhen die zugewiesene Bestrafung. Die Belohnungsfunktion bezüglich *Belief-States* wird, wie im Kapitel 4 beschrieben, mittels Erwartungswertbildung bestimmt. Eine Zufallsverteilung, für die mit Wahrscheinlichkeit 0.5 das Ziel erreicht ist, allerdings auch mit Wahrscheinlichkeit 0.5 eine Kollision auftritt, wird beispielsweise mit -20 bewertet.

Abb. 6.4 zeigt die Umgebungsbedingungen, wie sie für die Tests in diesem Kapitel verwendet werden. Die Roboter können sich in einem Feld der Größe $[10 \times 10]$ bewegen. Die grau hinterlegten Blöcke entsprechen Hindernissen. Die rote Region rechts unten ist die zu erreichende Zielregion. Je nach Startposition muss also zunächst Hindernis 1 umfahren werden, um dann die enge Passage zwischen Hindernis 2 und 3 zu passieren, die zur Zielregion führt. Die beiden Landmarken, die für Entfernungsmessungen verwendet werden, sind außerhalb des gezeigten Bereichs bei $[-5, 5]$ bzw. $[5, -5]$ angebracht. Diese Messungen können auch ausgeführt werden, wenn der direkte Weg von Hindernissen blockiert ist.

Die Interaktion der Roboter wird von der Belohnungsfunktion $R_G(b)$ beschrieben, die über dem gemeinsamen *Belief-Space* definiert ist. Entsprechend der Funktion R_L müsste diese Funktion die Wahrscheinlichkeit einer Kollision für gegebene Zustandsverteilung berücksichtigen. Da diese Wahrscheinlichkeit allerdings aufwendig zu berechnen ist, wird stattdessen eine Approximation durch die paarweise Summe der Mahalanobis-Distanzen verwendet. Die Mahalanobis-Distanz ist über den Abstand der Mittelwerte, sowie die Summe der Kovarianzen bestimmt, d.h.

$$M(\underline{x}^1, \underline{x}^2) = (\underline{x}^1 - \underline{x}^2)^T (\Sigma^1 + \Sigma^2)^{-1} (\underline{x}^1 - \underline{x}^2).$$

Diese Distanz berücksichtigt neben dem quadratischen Abstand auch Größe und Ausrichtung der Unsicherheitsmatrizen, wie Abb. 6.2 anhand verschiedener Verteilungspaare zeigt. Wie hier zu erkennen ist, entspricht ein kleiner Wert der Mahalanobis-Distanz einer hohen Wahrscheinlichkeit für eine Kollision. Die Bestrafungsfunktion wird schließlich folgendermaßen bestimmt:

$$R_G(\underline{x}) = -40 \cdot \sum_{(\underline{x}^i, \underline{x}^j), i \neq j} max(0, 1 - \frac{M(\underline{x}^i, \underline{x}^j)}{10}).$$

Mahalanobis-Distanzen > 10 werden somit ignoriert, sichere Kollisionen mit Distanz 0 werden wie statische Hindernisse mit -40 bewertet. Zwischen 0 und 10 nimmt die Bestrafung linear

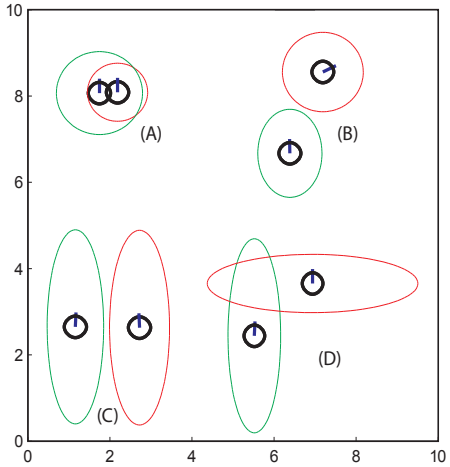


Abbildung 6.2: Die Abbildung zeigt verschiedene Kombinationen von normalverteilten Zustandsschätzungen, die jeweils über Mittelwert und Kovarianzmatrix bestimmt sind. Die zugehörigen Mahalanobis-Distanzen der vier Kombinationen sind: $(A) : 1, (B) : 20, (C) : 12, (D) : 5$.

mit der Distanz ab. Dies bedeutet für die dargestellten Beispiele: Keine Bestrafung für (B) und (C), (A) wird ein Wert von -36 und (D) ein Wert von -20 zugewiesen.

6.1.4 POMDP-Lösung

Um das in Kapitel 5 vorgestellte Verfahren einzusetzen, wird der Raum der Wahrscheinlichkeitsverteilungen auf Normalverteilungen projiziert. Durch Einsatz des *Extended Kalmanfilters* wird diese Projektion allerdings implizit durch die Prädiktion und Filterung ausgeführt. Die Auswahl der verwendeten *Belief-Points* folgt weitgehend dem Ansatz, den Brooks in [BMDW06] vorstellt, wobei ein äquidistantes Gitter im Parameterraum verwendet wird. In den hier beschriebenen Tests wird zusätzlich die Ausrichtung ϕ des Roboters sowie die Kreuzkovarianz σ^{xy} berücksichtigt.

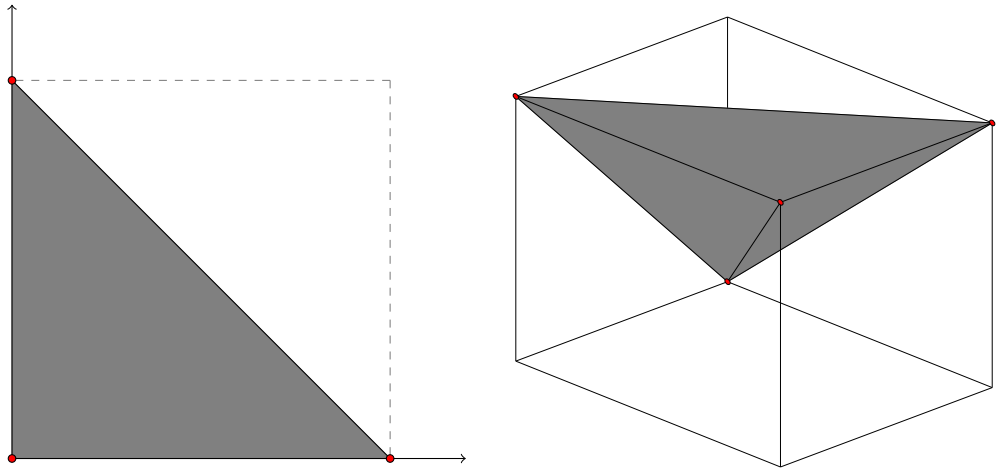
Insgesamt resultiert daraus ein sechsdimensionaler *Belief-Space* mit drei Dimensionen für den Mittelwert der Pose $[x, y, \phi]^T$, also Position und Ausrichtung der Roboter. Für die Beschreibung der Unsicherheit verbleiben drei Dimensionen für σ_x, σ_y und σ_{xy}. Um die Dimension niedrig zu halten, wird die Unsicherheit der Ausrichtung ϕ in der Planung nicht berücksichtigt, d.h. die Gütefunktion einer bestimmten Verteilung wird über den Winkel marginalisiert. Da die Messung der Ausrichtung ϕ_k als sehr genau vorausgesetzt wird, lässt diese Approximation allerdings keine großen Fehler erwarten.

Aufgrund besserer Interpolationseigenschaften wird die Darstellung der Unsicherheit von (σ_x, σ_y, σ_{xy}) transformiert zu $(e_1, \frac{e_2}{e_1}, \alpha_{e_1})$, wobei e_1 und e_2 die Eigenwerte der Kovarianzmatrix sind, d.h. die Länge der Hauptachsen der Unsicherheitsellipse und α_{e_1} der Winkel des größeren Eigenvektors, also die Ausrichtung der Unsicherheitsellipse. Der Zustandsvektor des transformierten *Belief-Space-MDP* hat somit folgende Form:

$$(x, y, \alpha, e_1, \frac{e_2}{e_1}, \alpha_{e_1}).$$

Aus diesem kontinuierlichen Raum müssen nun diskrete Punkte ausgewählt werden, die zur *Fitted Value Iteration* verwendet werden. Dafür wird ein äquidistantes Gitter über $\mathcal{B}$ verwendet. Die x- und y-Position wird dabei jeweils im Intervall $[0, 10]$ mit einer Auflösung von 0.5 diskretisiert. Der größere Eigenwert e_1 wird durch Werte in $\{0.05, 0.25, 0.55\}$ beschrieben. Das Verhältnis von größerem zu kleinerem Eigenwert, das die Form der Unsicherheitsellipse beschreibt, wird ebenfalls durch $\{0.05, 0.25, 0.55\}$ diskretisiert. Sowohl die Ausrichtung des Roboters als auch der Winkel des größeren Eigenwerts wird in $\frac{\pi}{4}$ rad-Schritte unterteilt. Insgesamt ergeben sich somit $20 \cdot 20 \cdot 8 \cdot 3 \cdot 2 \cdot 4 = 76800$ Punkte, an denen die Gütefunktion berechnet wird. Für die Interpolation wird *Freudenthal-Interpolation* verwendet mit der für äquidistante Gitter schnell die Interpolationskoeffizienten bestimmt werden können.

Freudenthal-Interpolation



(a) zweidimensionaler Raum (b) dreidimensionaler Raum

Abbildung 6.3: *Freudenthal-Interpolation.* Im zweidimensionalen Raum wird ein Würfel von zwei Simplexen ausgefüllt, im dreidimensionalen Raum von sechs. Die Eckpunkte des beinhaltenden Simplex dienen zur Interpolation

Bei der *Freudenthal-Interpolation* [Dav97] handelt es sich um ein Verfahren, das jeden Funktionswert einer d-dimensionalen reellen Funktion als lineare Kombination der Funktionswerte an $d + 1$ Gitterpunkten darstellt. Diese Gitterpunkte sind in jeder Dimension äquidistant verteilt. Für die Interpolation wird der d-dimensionale Definitionsbereich der Funktion in Würfel zwischen den Gitterpunkten unterteilt. Jeder dieser Würfel wird dann weiter in $d!$ Simplexe unterteilt. Liegt ein Punkt innerhalb eines Simplex, so wird dessen Funktionswert über die Eckpunkte des Simplex interpoliert. Um zu berechnen, in welchem Simplex ein Punkt zu liegen kommt, müssen zunächst die Koordinaten bezüglich des beinhaltenden Würfels bestimmt werden. Werden diese Würfel-Koordinaten aufsteigend sortiert, ergibt sich direkt das zugehörige

Simplex, da das Gebiet $(x_{i1} \geq x_{i2} \geq \dots)$ für jede Permutation der Indizes einem solchen Simplex entspricht. Im zweidimensionalen Fall entsprechen diese Gebiete Dreiecken, im dreidimensionalen Fall Tetraedern. Das Verfahren lässt sich aber auch für höhere Dimensionen in gleicher Weise anwenden. Nachdem bestimmt wurde, welche Eckpunkte das entsprechende Simplex aufspannen, kann der gegebene Punkt als konvexe Kombination dieser Punkte dargestellt werden. Die Koeffizienten dieser Repräsentation entsprechen direkt den Interpolationskoeffizienten.

Vorteil dieses Verfahrens ist zum einen, dass eine stetige, stückweise lineare Funktion entsteht, deren Funktionswerte an den Gitterpunkten der tatsächlichen Funktion entsprechen. Zum anderen lässt sich das Verfahren effizient implementieren, da die Berechnung der für die konvexe Kombination verwendeten Punkte und deren Koeffizienten in konstantem Aufwand $\mathcal{O}(d \log(d))$ möglich ist. Dies entspricht dem Aufwand der Sortierung der Würfel-Indizes.

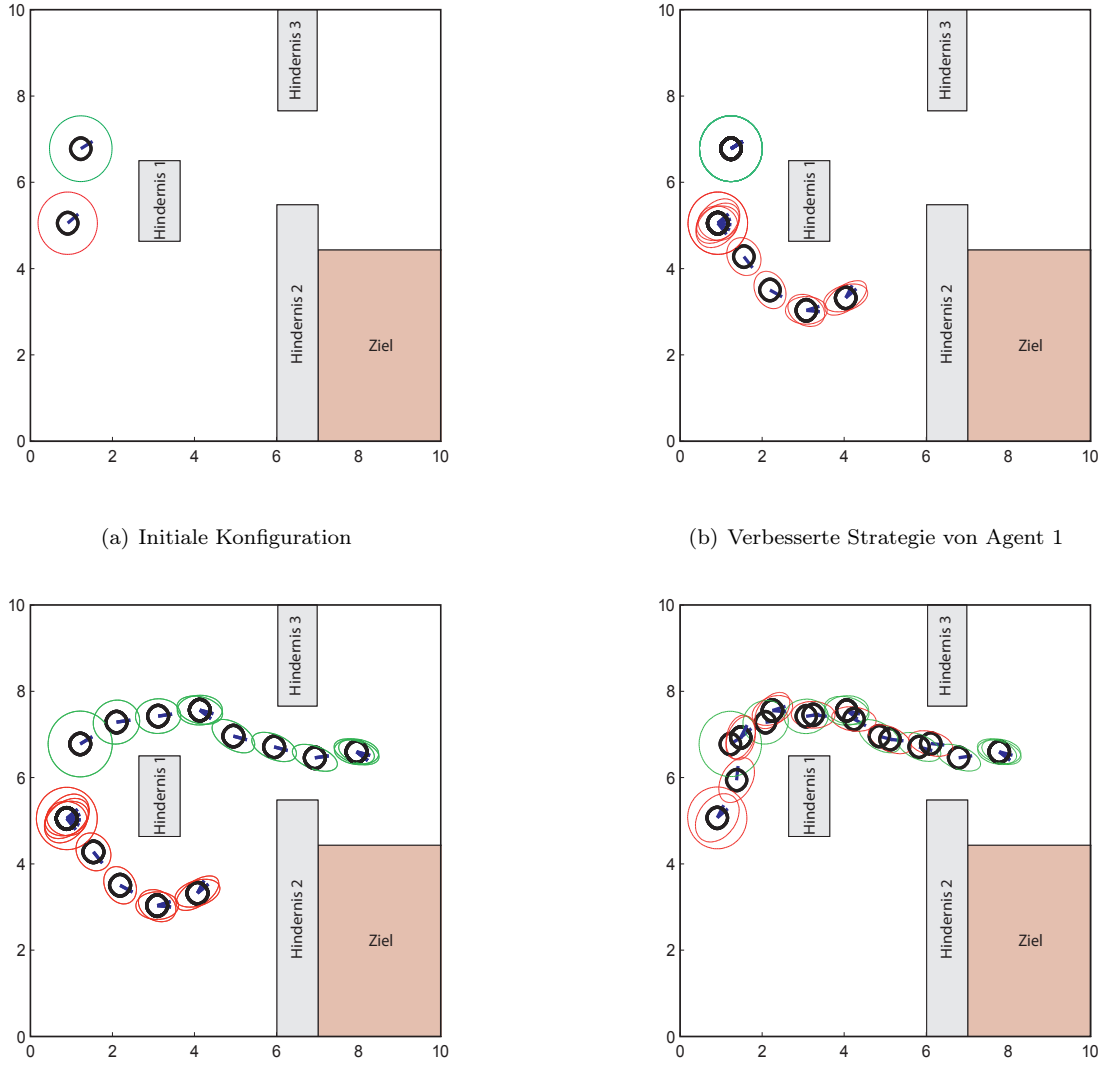
6.2 Beispiel für Planung

Um die Planung in der eben vorgestellten Testumgebung zu veranschaulichen, ist in Abb. 6.4 ein typischer Ablauf eines Planungsschrittes gezeigt. Die zwei Roboter R1 (rot) und R2 (grün) starten jeweils nach oben ausgerichtet. Der optimale Pfad verläuft daher für beide über Hindernis 1, hinter dem sie anfangs positioniert sind. Die Unsicherheit der Roboter ist jeweils durch die 3σ-Ellipsen repräsentiert. Der initiale gemeinsame Plan sieht vor, dass beide Roboter über den gesamten Planungshorizont stehen bleiben.

Im ersten Schritt wird nun für R1 die optimale Strategie unter gegebenen Voraussetzungen bestimmt. So kann er nicht den bevorzugten Weg oberhalb des ersten Hindernisses wählen, da dort R2 den Weg versperrt und aufgrund der unveränderten initialen Strategie stehen bleibt. Als Alternative wird der in Bild (b) gezeigte Plan bestimmt, der in dieser Situation für R1 optimal ist, da er die Strategie von R2 nicht verändern kann. Als nächstes wird für R2 geplant, indem die eben berechnete Strategie von R1 festgehalten wird. R2 wählt nun den in Bild (c) gezeigten Pfad, da dieser nicht von einem anderen Roboter blockiert ist.

Da beide Roboter in dieser ersten Planungsphase ihre Strategie änderten, wird ein weiterer Durchgang ausgeführt. Somit wird wiederum die Strategie des ersten Roboters optimiert. Nun gelten allerdings im Gegensatz zur vorherigen Planung andere Bedingungen. Der Plan von R2 sieht vor, dass dieser seine aktuelle Position in Richtung Ziel verlässt. Damit ist auch für R1 Platz, um den gewünschten Pfad oberhalb des Hindernisses einzuschlagen. Nun wird erneut für R2 geplant, allerdings hat sich für diesen die Situation nicht verändert. Somit verfolgen beide Roboter die optimale Strategie und das Verfahren terminiert.

Hatten beide Roboter die gleiche Information über die initiale Zustandsverteilung, so kommen beide unabhängig voneinander zum gleichen Ergebnis. Die erste Aktion der somit bestimmten gemeinsamen Strategie wird im Folgenden von den Robotern ausgeführt und gegebenenfalls die Zustandsschätzung durch Messupdates aktualisiert. Schließlich wird der bisher beschriebene Planungsschritt wiederholt, bis die Roboter das Ziel erreicht haben.



(a) Initiale Konfiguration

(b) Verbesserte Strategie von Agent 1

(c) Verbesserte Strategie von Agent 2

(d) Verbesserte Strategie von Agent 1

Abbildung 6.4: Beispiel für einen Schritt der Planung. (a) Zwei Roboter R1 (rot) und R2 (grün) starten hinter Hindernis 1 (b) Zunächst wir die beste Stellgrößenabfolge für R1 bestimmt. (c) Nun wird für R2 geplant, der seinen gewünschten Weg oberhalb Hindernis 1 enschlagen kann. (d) Durch erneute Planung für R1 kann nun der gewünschte Pfad oberhalb des Hindernis eingeschlagen werden, da R2 seine Position im nächsten Schritt verlassen wird.

6.3 Referenzverfahren

Als Referenzverfahren (RV) für die folgenden Tests wird ein einfaches Brute-Force-Maximierungsverfahren eingesetzt. Innerhalb des Planungshorizonts $\tilde{H}$ wird im Gegensatz zur alternierenden Optimierung (AO) der komplette Entscheidungsbaum für Kombinationen von Aktionen der Agenten aufgestellt. Jeder Pfad dieses Baumes entspricht einem bestimmten gemeinsamen Verhalten, für das die erwartete Güte jedes Agenten berechnet werden kann. Schließlich wird die Summe der erwarteten Güte jedes Agenten gebildet und über diese Summe maximiert.

Für N Agenten und Planungshorizont $\tilde{H}$ gibt es $|\mathcal{U}|^{\tilde{H} \cdot N}$ Kombinationen von Aktionenfolgen, für die jeweils die erwartete Güte bestimmt werden muss. Diese Berechnung stößt aufgrund Beschränkungen der Rechenleistung schon bei kleinem Planungshorizont schnell an seine Grenzen. So kann das RV in der hier verwendeten Implementierung mit maximal Horizont $\tilde{H} = 2$ eingesetzt werden. Auf das Pfadplanungsproblem angewandt würde diese Planung allerdings den positiven Einfluss der Zielregion nicht mit einbeziehen, wenn das Ziel nicht innerhalb $\tilde{H}$ Schritten erreichbar ist.

Deshalb wird zusätzlich die Güte der Zustandsverteilung $\underline{b}^i_{\tilde{H}}$ mit Hilfe der lokalen Gütefunktion J^i_L bewertet, die approximativ über einen größeren Horizont berechnet wird. Insgesamt wird somit für jede mögliche gemeinsame Aktionenfolge die Funktion

$$J_{RV}(\underline{u}^{1,\dots,N}_{0,\dots,\tilde{H}-1}, \underline{b}_0) = \mathrm{E}\left\{ \sum_{k=0}^{\tilde{H}-1} \Big[\sum_{i=1}^{N} (R_L(\underline{\boldsymbol{x}}^i_k, \underline{u}_k)) + R_G(\underline{\boldsymbol{x}}_k, \underline{u}_k)) \Big] \right\} + \sum_{i=1}^{N} (J^i_L(\underline{b}^i_{\tilde{H}}))$$

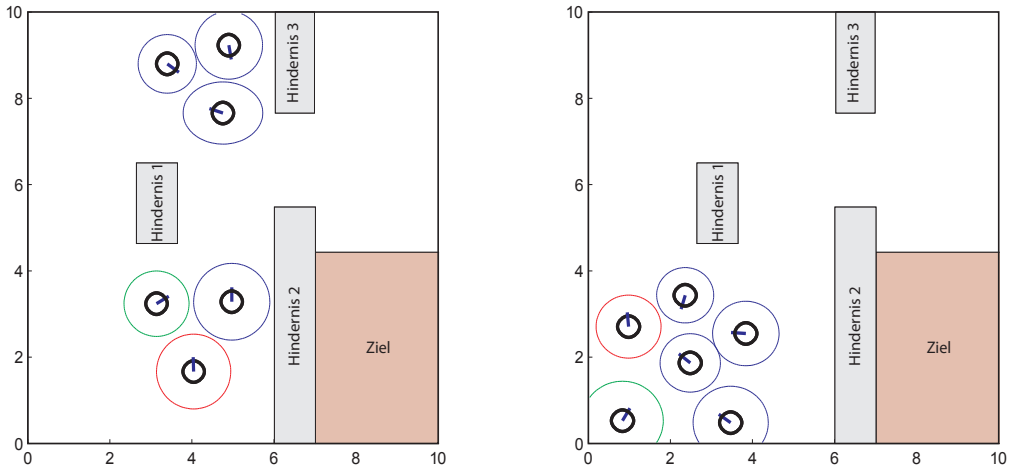
ausgewertet. Der erste Summand beschreibt die Summe der erwarteten Gütefunktion einer bestimmten Kombination von gemeinsamen Aktionenfolgen über den Horizont $\tilde{H}$. Der zweite Summand bewertet die erwartete lokale Güte, welche die Agenten von Verteilung $\underline{b}_{\tilde{H}}$ aus erreichen können. Die globale Gütefunktion, die Interaktionen zwischen den Agenten beschreibt, taucht allerdings nur im ersten Summanden auf. Somit werden Interaktionen nur bis Horizont $\tilde{H}$ betrachtet. Schließlich wird über alle möglichen Aktionenfolgen maximiert und die optimale Strategie ausgewählt:

$$RV(\underline{b}_0) = \arg \max_{\underline{u}^{0,\dots,N}_{1,\dots,\tilde{H}-1}} J_{RV}(\underline{u}^{0,\dots,N}_{1,\dots,\tilde{H}-1}).$$

6.4 Testszenarien

In den folgenden zwei Abschnitten werden Ergebnisse verschiedener Tests vorgestellt, die das neuartige Verfahren AO im Vergleich zu einem einfachen Referenzverfahren zeigen. Zunächst wird die Qualität der berechneten Strategien, also erhaltene Güte und Anzahl der Kollisionen verglichen. Schließlich wird die Laufzeit der einzelnen Verfahren mit einer unterschiedlichen Anzahl an Agenten sowie verschiedenen Planungshorizonten evaluiert.

Um gleiche Umgebungsbedingungen für alle Tests zu gewährleisten, wurde jeweils das gleiche System- und Beobachtungsmodell vorausgesetzt. Die Standardabweichung der Vorwärtsbewegung $\sqrt{\mathrm{Var}(\boldsymbol{v}^s)}$ wurde dabei auf 0.05 festgelegt, $\sqrt{\mathrm{Var}(\boldsymbol{v}^\phi)}$ auf 0.2 rad. Mit diesen Werten ist das verwendete Bewegungsmodell (6.1) bestimmt. Die Standardabweichungen der Distanzmessungen $\sqrt{\mathrm{Var}(\boldsymbol{w}^{d1})}$ sowie $\sqrt{\mathrm{Var}(\boldsymbol{w}^{d1})}$ wurden ebenfalls auf 0.5 gesetzt. Die Winkelmessung wurde mit $\sqrt{\mathrm{Var}(\boldsymbol{w}^\phi)} = 0.05$ rad als sehr genau vorausgesetzt, was die in Abschnitt 6.1.4 beschriebene Vereinfachung rechtfertigt.



(a) (b)

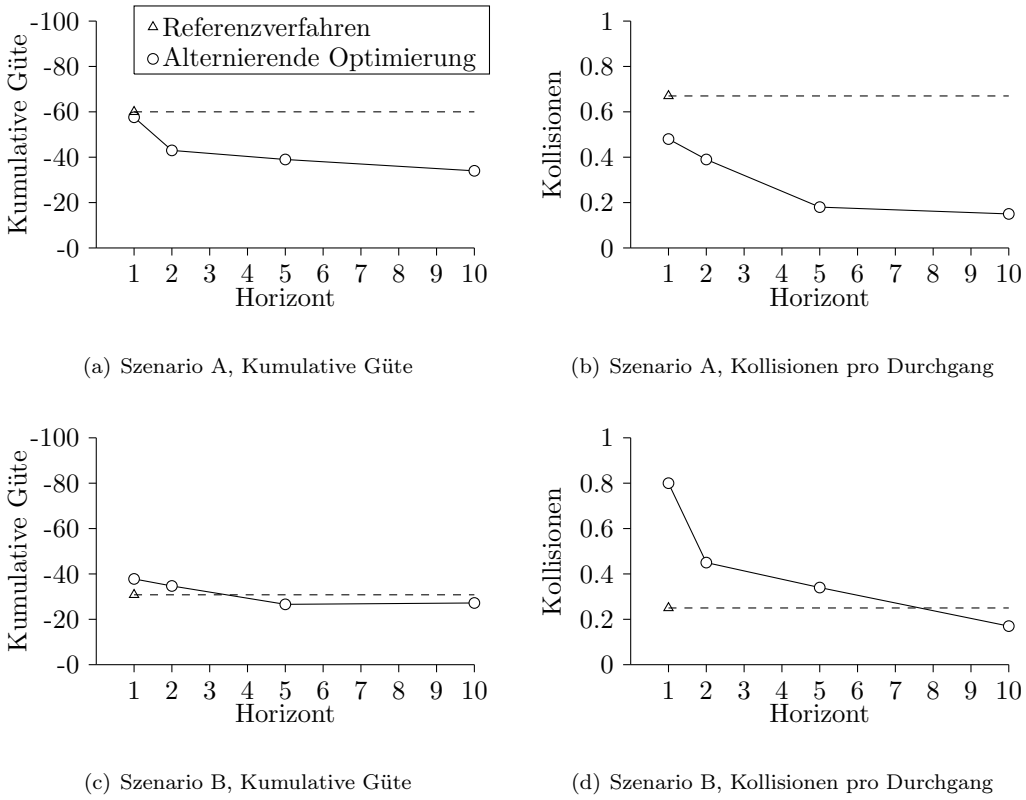
Abbildung 6.5: Diese Startszenarien wurden für die Tests in Abschnitt 6.4.1 verwendet. In beiden Fällen versuchen sechs Roboter, auf die rechte Seite zu kommen, wobei sie die enge Stelle zwischen Hindernis 2 und Hindernis 3 passieren müssen.

Sowohl AO als auch RV verwenden vorberechnete lokale Gütefunktionen, die jeweils über den Planungshorizont $H = 40$ berechnet werden. Wie in Abschnitt 5.7 beschrieben, kann der gemeinsame Planungshorizont $\tilde{H}$ bei AO kleiner als der lokale Planungshorizont sein, um eine effizientere Berechnung zu ermöglichen. So wird die Interaktion der Agenten in der Planungsphase nur für die nächsten $\tilde{H}$ Schritte berücksichtigt. In den folgenden Tests wird dieser gemeinsame Planungshorizont variiert.

6.4.1 Qualität der Lösung

Um die Qualität der gemeinsamen Strategie zu evaluieren, wurden die zwei in Abb. 6.5 gezeigten Startkonfigurationen mit verschiedenen Horizontlängen getestet. Für AO wurde dabei ein Horizont von $1, 2, 5$ und 10 eingesetzt, RV konnte aufgrund von Speicherplatzbeschränkungen für diese Anzahl von Agenten nur mit Horizont 1 berechnet werden. (Siehe Ergebnisse in Abschnitt 6.4.1). In beiden Szenarien sind die Roboter anfangs dicht gedrängt, so dass auf jeden Fall ein koordiniertes Verhalten notwendig ist.

In Abschnitt 5.8 wurde darauf eingegangen, wie Beobachtungen über andere Agenten verwendet werden können und das Verfahren als *Open-Loop*-Approximation zu verstehen ist. Die Tests wurden daher mit zwei verschiedenen Beobachtungsszenarien durchgeführt. Zunächst wurde eine zentrale Messeinrichtung vorausgesetzt, von der alle Agenten die gleichen Daten erhalten. Da somit garantiert ist, dass auch alle Agenten die gleiche gemeinsame Strategie finden, können auch die jeweiligen Aktionen zur Prädiktion eingesetzt werden. Die Ergebnisse dieser Tests ist in Abb. 6.6 gezeigt.



(a) Szenario A, Kumulative Güte

(b) Szenario A, Kollisionen pro Durchgang

(c) Szenario B, Kumulative Güte

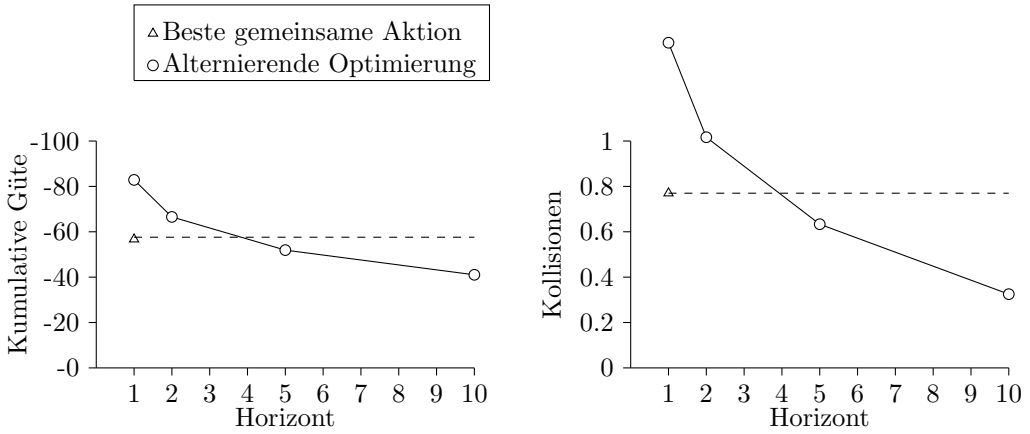
(d) Szenario B, Kollisionen pro Durchgang

Abbildung 6.6: Dargestellt sind die Ergebnisse der verschiedenen Algorithmen mit gemeinsamen Beobachtungen. Die Kreise stehen für die Ergebnisse der *AO* mit unterschiedlichem gemeinsamem Planungshorizont, die Dreiecke für die Ergebnisse des *RV*. Die gestrichelte Linie dient nur zur besseren Lesbarkeit, da *RV* ausschließlich mit Horizont eins getestet wurde.

Als realistischeres Messszenario wurden unabhängige Sensoren für jeden Roboter vorausgesetzt. Diese unterlagen jeweils unabhängigem Rauschen, so dass nicht mehr die gleiche Informationsgrundlage gegeben ist. Da nun nicht mehr garantiert ist, dass alle Roboter die gleiche gemeinsame Strategie berechnen, können auch die berechneten Aktionen der anderen Agenten nicht mehr zur Zustandsschätzung verwendet werden. Anstatt eines Prädiktionsschrittes wurde daher die Unsicherheit in alle Richtungen vergrößert, indem auf die Kovarianzen σ_x und σ_y jeweils Standardabweichung 1 addiert wurde. Die Ergebnisse dieses Tests sind in Abb. 6.7 dargestellt.

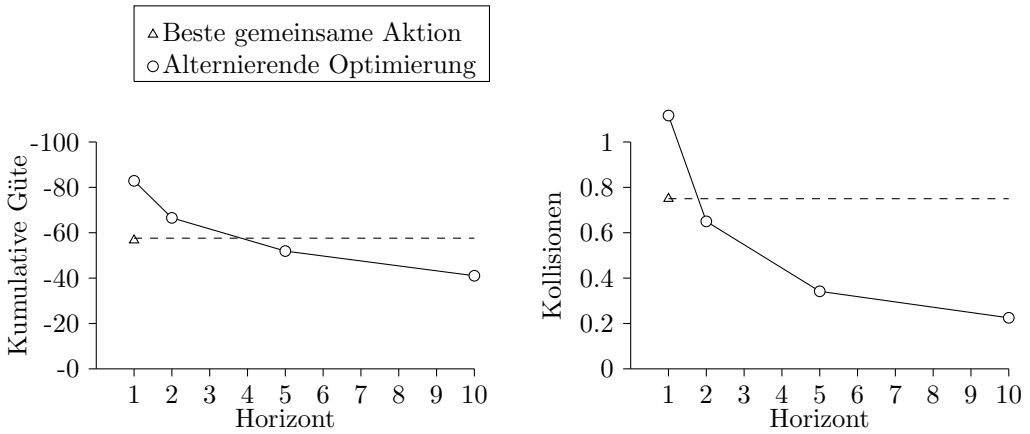
Die Ergebnisse sind jeweils die durchschnittlichen Werte aus 50 Durchläufen von der gleichen Startkonfiguration. Links ist jeweils die kumulative Güte der Läufe dargestellt, die durch die verwendete Belohnungsfunktion nur negative Werte annehmen kann. Zur besseren Lesbarkeit ist die negative Achse nach oben aufgetragen, weshalb kleine Werte in diesem Fall als besser zu bewerten sind. Rechts ist die durchschnittliche Anzahl an Kollisionen pro Durchlauf angegeben. Die gestrichelte Linie zeigt die Güte des Referenzverfahrens, das nur für Horizont 1 berechnet werden konnte.

In den Tests ist zu sehen, dass sich das globale Optimum meist besser verhält als die entsprechende *AO* über gleichen Planungshorizont. Mit zunehmendem Planungshorizont verbessert sich die Qualität der Planung allerdings deutlich, so dass außer bei einem Test (6.6d) schon



(a) Szenario A,Kumulative Güte

(b) Szenario A, Kollisionen pro Durchgang



(c) Szenario B, Kumulative Güte

(d) Szenario B, Kollisionen pro Durchgang

Abbildung 6.7: Dargestellt sind die Ergebnisse der verschiedenen Algorithmen mit jeweils unabhängigen Messungen. Die Kreise stehen für die Ergebnisse der *AO* mit unterschiedlichem gemeinsamem Planungshorizont, die Dreiecke für die Ergebnisse des *RV*. Die gestrichelte Linie dient nur zur besseren Lesbarkeit, da *RV* nur mit Horizont eins getestet wurde.

Horizont 5, in jedem Fall aber Horizont 10 bessere Ergebnisse erzielt. Das *RV* konnte schon Horizont 2 bei 6 Agenten aufgrund von Speicherplatzbeschränkungen nicht mehr berechnen.

Der Vergleich zwischen Abb. 6.6 und 6.7 zeigt, dass bessere Resultate erreicht werden, wenn den Agenten die gleiche Information zur Verfügung steht. Dies ist nicht überraschend, da in diesem Fall alle Agenten garantiert die gleiche gemeinsame Strategie berechnen und ausführen. Aber auch im realistischeren Fall, in dem jedem Agenten unterschiedliche Beobachtungen vorliegen, können mit *AO* deutlich bessere Ergebnisse als mit dem *RV* erzielt werden.

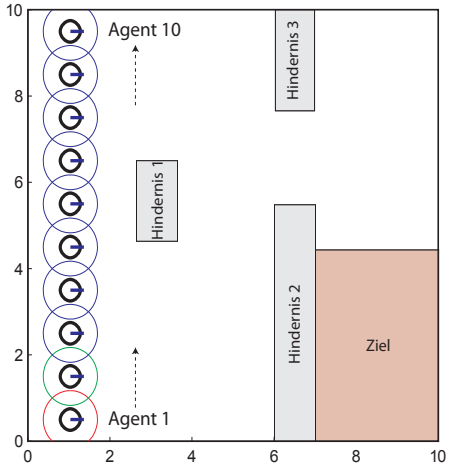


Abbildung 6.8: Laufzeittest: Als initiale Position werden die Roboter von unten links beginnend nach oben aufgereiht.

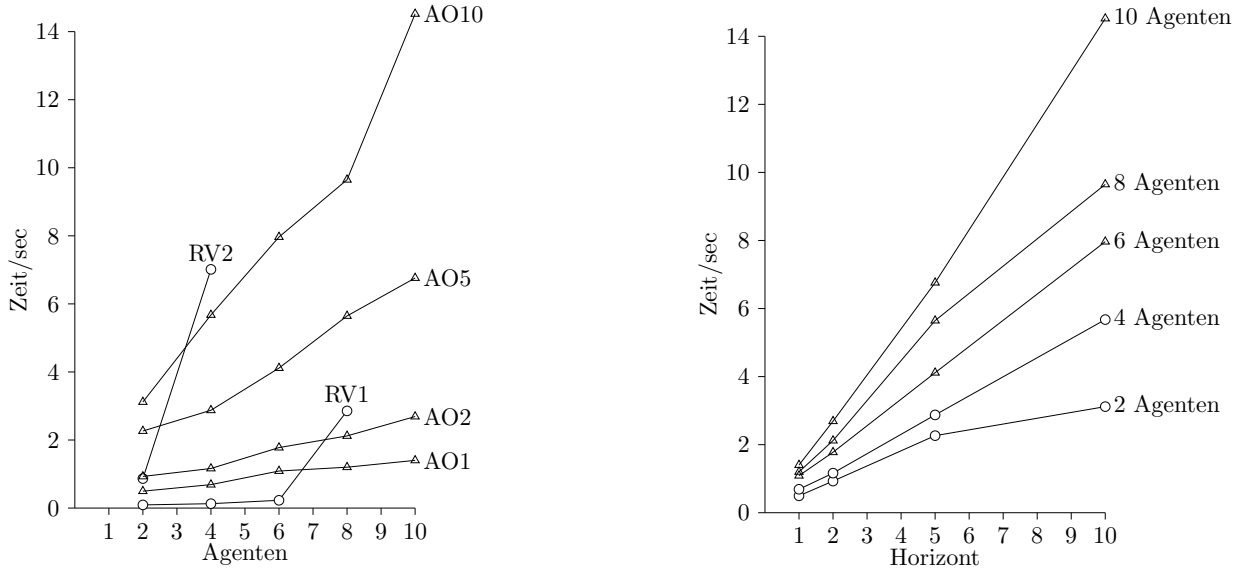
6.4.2 Laufzeit

Um die Skalierbarkeit bezüglich der Anzahl der Agenten sowie bezüglich des Planungshorizontes zu demonstrieren, sind im Folgenden Tests beschrieben, welche die Laufzeit der verschiedenen Algorithmen im Vergleich zeigen.

Um eine unterschiedliche Anzahl an Agenten vergleichbar zu testen, wurde die in Abb. 6.8 gezeigte initiale Konfiguration gewählt. Hier werden die Agenten beginnend vom unteren linken Rand nach oben aufgereiht. Die Mittelwerte der initialen Zustandsschätzungen liegen somit bei N Agenten auf den Punkten $[1, 0.5]^T$ bis $[1, 0.5 + (N-1)]^T$. In Abb. 6.9 sind die durchschnittlichen Berechnungszeiten eines Planungsschrittes für verschiedene Einstellungen dargestellt.

Diagramm (a) zeigt die Laufzeit in Abhängigkeit der Anzahl an Agenten. Das Referenzverfahren mit gemeinsamem Planungshorizont 1 konnte bis zu 8 Agenten bewältigen, bei Horizont 2 konnte nur noch ein Szenario mit 3 Agenten berechnet werden. Höhere Werte überstiegen den verfügbaren Speicherplatz der verwendeten Implementierung. AO konnte hingegen für 10 Agenten mit Planungshorizont 10 planen. Das RV berechnet die Lösung für eine kleine Anzahl an Agenten noch schneller als AO mit gleichem Planungshorizont. Für eine größere Anzahl an Agenten kann allerdings AO schneller eine gemeinsame Strategie berechnen. Insgesamt zeigt sich, dass sich die Laufzeit der AO in den hier vorgestellten Tests in etwa linear mit der Anzahl an Agenten verhält.

Die gleichen Ergebnisse sind in Diagramm (b) mit jeweils konstanter Anzahl an Agenten aufgetragen. Hier sind allerdings nur die Werte der AO angegeben, da von RV nur bis maximal Horizont 2 Ergebnisse verfügbar sind. Auch hier ist zu sehen, dass sich die Laufzeit entgegen dem Worst-Case-Fall in etwa linear mit steigendem Horizont verhält.



(a) Laufzeit gegen # Agenten

(b) Laufzeit gegen Horizont

Abbildung 6.9: Die linke Abbildung zeigt *AO* für eine verschiedene Anzahl von Agenten. Die Startkonfigurationen sind in Abb. 6.8 dargestellt. Hier ist zusätzlich die Laufzeit des Referenzverfahrens eingezeichnet. Die rechte Abbildung zeigt die Ergebnisse der gleichen Tests, allerdings gegen die Anzahl der Agenten aufgetragen.

KAPITEL 7

Zusammenfassung und Ausblick

In dieser Arbeit wurde ein neuartiges Regelungsverfahren für Multi-Agenten-Systeme mit nicht direkt zugänglichen Zuständen vorgestellt. Allgemein lassen sich Systeme dieser Art als *POSG* modellieren, allerdings ist die optimale Lösung dieser Probleme zu komplex, um optimale Strategien in realistischen Anwendungen zu finden. Daher wurde zunächst ein eingeschränktes Modell vorgestellt, das Transitions- und Beobachtungsunabhängigkeit der einzelnen Agenten und eine additive Belohnungsfunktion voraussetzt. Somit lässt sich das Problem in Teilprozesse für jeden Agenten aufteilen, die weitgehend unabhängig voneinander betrachtet werden können. Dies reduziert die Komplexität drastisch, allerdings schränken die Voraussetzungen auch die möglichen Anwendungsgebiete ein. Es gibt jedoch interessante Anwendungen, wie z.B. die Pfadplanung in einer teilweise beobachtbaren Umgebung, die sich mit diesem Modell beschreiben lassen.

Das vorgestellte Verfahren baut auf Ansätzen für die Lösung ähnlicher Probleme auf (*ND-POMDPs* [NVTY05, VMY+07]), die allerdings einen diskreten Zustandsraum voraussetzen. Im Gegensatz dazu lässt das im Rahmen dieser Arbeit entwickelte Verfahren auch die Regelung von Systemen mit kontinuierlichem Zustandsraum zu. Dafür wurde die Idee der alternierenden Optimierung mit bestehenden Ansätzen für die Lösung von lokalen POMDPs über kontinuierlichem Zustandsraum [BMDW06] kombiniert. Anhand von Experimenten wurde gezeigt, dass das Verfahren erfolgreich auf Probleme aus der Multi-Roboter-Pfadplanung angewendet werden kann. Für Beispiele dieser Art lassen sich einfache parametrische Darstellungen der vorkommenden Dichten finden, was Voraussetzung für dessen Einsatz ist. Mit Hilfe des entwickelten Verfahrens konnten im Gegensatz zur Brute-Force-Maximierung der Gütefunktion weitaus mehr Agenten und ein höherer Planungshorizont betrachtet werden.

Als Erweiterung des Verfahrens wäre denkbar, die globale Belohnungsfunktion ähnlich wie bei *ND-POMDP*s weiter zu unterteilen. In dieser Arbeit wurde angenommen, dass diese globale Funktion vom Zustand aller Agenten abhängt. Ist es aufgrund der betrachteten Problemstruktur möglich, diese in die Summe mehrerer Funktionen zu unterteilen, die jeweils nur von einem Teil der Agenten abhängen, so könnte die Effizienz des Verfahrens weiter gesteigert werden.

ANHANG A

Anhang

A.1 MDP-Verteilung des Zustands

Initiale Verteilung

$$f_{\underline{x}_0}(\underline{x}_0) = \delta(\underline{x}_0)$$

Rekursive Berechnung

$$\begin{aligned} f_{\underline{x}_{k+1}}(\underline{x}_{k+1}) &= p(\underline{x}_{k+1}|\pi) \\ &= \int_{\mathcal{X}} p(\underline{x}_{k+1}, \underline{x}_k|\pi) d\underline{x}_k \\ &= \int_{\mathcal{X}} p(\underline{x}_{k+1}|\underline{x}_k, \pi) p(\underline{x}_k|\pi) d\underline{x}_k \\ &= \int_{\mathcal{X}} T(\underline{x}_{k+1}|\underline{x}_k, \mu_k(\underline{x}_k)) f_k(\underline{x}_k) d\underline{x}_k. \end{aligned}$$

A.2 Unabhängige Zustandsverteilungen

Die lokale Repräsentation der globalen Belohnungsfunktion setzt unabhängige Zustandsverteilungen der Agenten voraus. Im Folgenden wird gezeigt, dass sich bei initialer Unabhängigkeit diese Eigenschaft auch mit beliebigen Prädiktions- und Filterschritten nicht ändert.

Voraussetzung

$$p(\underline{x}_k) = \prod_i p(\underline{x}_k^i)$$

Prädiktion

$$\begin{aligned}
p(\underline{x}_{k+1}) &= \int_X p(\underline{x}_{k+1}, \underline{x}_k) d\underline{x}_k \\
&= \int_X p(\underline{x}_{k+1}|\underline{x}_k) p(\underline{x}_k) d\underline{x}_k \\
&= \int_{X^1} \dots \int_{X^N} \prod_i p(\underline{x}^i_{k+1}|\underline{x}^i_k) \prod_i p(\underline{x}^i_k) d\underline{x}^N_k \dots d\underline{x}^1_k \\
&= \int_{X^1} \dots \int_{X^N} \prod_i p(\underline{x}^i_{k+1}, \underline{x}^i_k) d\underline{x}^N_k \dots d\underline{x}^1_k \\
&= \prod_i p(\underline{x}^i_{k+1})
\end{aligned}$$

Filterung

$$\begin{aligned}
p(\underline{\boldsymbol{x}}_k|\underline{z}_k) &= \frac{1}{p(\underline{z}_k)} p(\underline{z}_k, \underline{x}_k) \\
&= \frac{1}{p(\underline{z}_k)} p(\underline{z}_k|\underline{x}_k) p(\underline{x}_k) \\
&= \frac{1}{p(\underline{z}_k)} \prod_i p(z^i_k|\underline{x}^i_k) \prod_i p(\underline{x}^i_k) \\
&= \frac{1}{p(\underline{z}_k)} \prod_i p(z^i_k|\underline{x}^i_k) p(\underline{x}^i_k) \\
&= \frac{1}{p(\underline{z}_k)} \prod_i p(z^i_k, \underline{x}^i_k) \\
&= \frac{1}{p(\underline{z}_k)} \prod_i p(\underline{x}^i_k|\underline{z}^i_k) p(\underline{z}^i_k) \\
&= \prod_i p(\underline{x}^i_k|\underline{z}^i_k)
\end{aligned}$$

A.3 Backward Links

Nimmt die Belohnungsfunktion $\underline{r}_{G,k}$ nur negative Werte an, so kann die Gütefunktion in der Praxis noch schneller berechnet werden, wenn die Interaktion nur zeitlich oder örtlich lokal auftritt. Dafür wird die Funktion $\underline{j}_k$ für $k = 1, \dots, H$ unter der Bedingung $\underline{r}_{G,k} = 0$ vorberechnet. Nimmt nun $\underline{r}_{G,k}$ nur an bestimmten Stellen negative Werte an, so ändern sich die Einträge der Gütefunktion $\underline{j}_k$ ebenfalls nur an bestimmten Stellen.

Im Folgenden bezeichne j'_k die Gütefunktion, die entsteht, wenn für $\underline{r}_{G,k}$ auch Werte kleiner Null gelten. Unterscheidet sich der i-te Eintrag des Vektors $\underline{j}'_k$ von $\underline{j}_k$, so ist aus (5.4) abzulesen, dass entweder der i-te Eintrag von $\underline{r}_{G,k} \neq 0$ ist oder sich aber $\max_{u_k} A^{u_k}(i,:) \cdot j'_{k+1}$ im Vergleich zur ursprünglichen Funktion geändert hat. Hierbei bezeichnet $\mathbf{A}^{u_k}(i,:)$ die i-te Zeile der Matrix $\mathbf{A}^{u_k}$. Da aber $\underline{j}'_{k+1} < j_{k+1}$ für alle Einträge gilt, ist

$$\mathbf{A}^{u}(i,:) \cdot j'_{k+1} < \mathbf{A}^{u}(i,:) \cdot j_{k+1} \text{ für } u = \arg\max_{\underline{u}_k} \mathbf{A}^{\underline{u}_k}(i,:) \cdot j_{k+1}.$$

Wäre dies nicht der Fall, so hätte sich $\mathbf{A}^{\tilde{u}}(i,:) \cdot j'_{k+1}$ für ein $\tilde{u} \neq u$ vergrößert, was nach Voraussetzung aber nicht möglich ist.

Zusammengefasst heißt das, dass sich der i-te Eintrag von $\underline{j}_k$ nur ändert, wenn entweder der i-te Eintrag von $\underline{r}_{G,k} \neq 0$ ist oder $\underline{j}_{k+1} \neq j_k$ für Einträge, die in $\mathbf{A}^{u}(i,:) \neq 0$ sind (mit $\underline{u}$ wie eben definiert).

Diese $\mathbf{A}^{u}(i,:)$ können vorberechnet werden, da sie nicht von der abgeänderten Belohnungsfunktion abhängen und für jeden Zeitschritt zusammen in eine Matrix $\mathbf{A}_k^{\max}$ geschrieben werden. $\mathbf{A}_k^{\max}$ besteht also aus den Zeilen $\mathbf{A}^{u}(i,:) \neq 0)$ mit $\underline{u} = \arg\max_{\underline{u}_k} \mathbf{A}^{u_k}(i,:) \cdot j_{k+1}$. Nun kann induktiv gezeigt werden, dass sich ein Eintrag von $\underline{j}_{H-\xi}$ nur dann ändern kann, wenn der entsprechende Eintrag in

$$\sum_{k=0,\ldots,\xi} \Big(\prod_{l=0,\ldots,k-1} \mathbf{A}^{\max}_{H-\xi+l} \Big) r_{H-\xi+k}$$

ungleich 0 ist. Für $k = 0$ ist das Produkt über $0, \ldots, k-1$ als Einheitsmatrix definiert. Werden auch diese Matrizenprodukte vorberechnet, die sich nicht mit der Belohnungsfunktion ändern, so wird diese Berechnung von $\mathcal{O}(H|B|^2)$ Multiplikationen dominiert. Die Berechnung hat also die gleiche Komplexität wie die *Value-Iteration* selbst. Werden aber nur wenige $\underline{r}_{G,l}$ verändert, so ist in der Praxis ein deutlicher Geschwindigkeitsvorteil zu erkennen. In Abb. A.1 ist die zu Grunde liegende Idee anschaulich dargestellt.

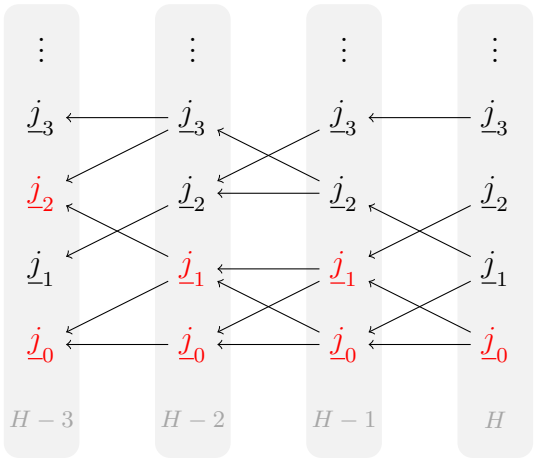


Abbildung A.1: Dargestellt ist ein Beispiel für *Value Iteration*. Die schwarzen Pfeile zeigen, welche Werte der Gütefunktion der jeweils besten Aktion entsprechen und somit für die Iteration verwendet werden. Verringert sich nun der Wert der Gütefunktion $\underline{j}_0$ im Horizont H, so sind nur die hervorgehobenen $\underline{j}_i$ betroffen. Für alle anderen Punkte kann sich die „beste“ Aktion nicht ändern, sie bleiben somit konstant.

Literaturverzeichnis

[AZ09] M. Allen and S. Zilberstein. Complexity of Decentralized Control: Special Cases. In *Proceedings of the Twenty-Third Neural Information Processing Systems Conference*, Vancouver, British Columbia, Canada, 2009.

[Bel57] R.E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, New Jersey, 1957.

[Ber95] D.P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, Massachusetts, 1995.

[BMDW06] A. Brooks, S. Makarenko, A. Williams, and H. Durrant-White. Parametric POMDPs for Planning in Countinuous State Spaces. In *Robotics and Autonomous Systems*, volume 54, pages 887 – 897, 2006.

[BZI00] D.S. Bernstein, S. Zilberstein, and N. Immerman. The Complexity of Decentralized Control of Markov Decision Processes. In *Mathematics of Operations Research*, page 2002, 2000.

[Dav97] S. Davies. Multidimensional Triangulation and Interpolation for Reinforcement Learning. In *Neural Information Processing Systems 9*, 1997.

[EMGST04] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate Solutions for POSGs with Common Payoffs. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

[EMGST05] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Game Theoretic Control for Robot Teams. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005.

[GL06] A.Y. Guo and V. Lesser. Stochastic Planning for Weakly Coupled Distributed Agents. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 326–328, 2006.

[GM08] J. Goldsmith and M. Mundhenk. Competion Adds Complexity. In *Advances in Neural Information Processing Systems 20*, 2008.

[HBZ04] E.A. Hansen, D.S. Bernstein, and S. Zilberstein. Dynamic Programming for POSGs. *Proceedings of the 19th national conference on Artificial Intelligence*, 2004.

[HR09] Ruijie He and Nicholas Roy. Efficient POMDP Forward Search by Predicting the Posterior Belief Distribution, 2009.

[KLC98] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artif. Intell.*, 101:99–134, 1998.

[KSV05] J.R. Kok, M.T.J. Spaan, and N. Vlassis. Non-communicative multi-robot coordination in dynamic environments. *Robotics and Autonomous Systems*, 50:2–3, 2005.

[MHC08] S.A. Miller, Z.A. Harris, and E.K.P. Chong. A POMDP Framework for Coordinated Guidance of Autonomous UAVs for Multitarget Tracking. *EURASIP J. Adv. Signal Process*, 2009:1–17, 2008.

[NTY+03] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and Marsella S. Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings. In *In IJCAI*, 2003.

[NVTY05] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Network Distributed POMDPs: A Synthesis of Distributed Contstraint Optimization and POMDPs. In *AAAI*, 2005.

[OSV08] F.A. Oliehoek, M.T.J. Spaan, and N. Vlassis. Optimal and Approximate Q-Value Function for Dec-POMDPs. *J. Artif. Int. Res.*, 32:289–353, 2008.

[OSW08] F.A. Oliehoek, M.T. Spaan, and S. Whiteson. Exploiting Locality of Interaction in Factored Dec-POMDPs. In *Procedings of 7th International Conference on Autonomous Agents and Multiagent Systems*, 2008.

[Pou02] C. Poupart, P.and Boutilier. Value-directed Compression of POMDPS. In *In NIPS 15*, 2002.

[Sim06] D. Simon. *Optimal State Estimation:Kalman, H Infinity, and Nonlinear Approaches*. Wiley & Sons, 2006.

[SZ05] F.; Szer, D.; Charpillet and S. Zilberstein. MAA*: A Heuristic Search Algorithm for Solving Decentralized POMDPs. In *In Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence.*, 2005.

[VMY+07] P. Varakantham, J. Marecki, Y. Yabu, M. Tambe, and M. Yokoo. Letting Loose a SPIDER on a Network of POMDPs: Generating Quality Guaranteed Policies. In *AAMAS*, 2007.

[vNM44] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1953 edition, 1944.

[VNTY06] P. Varakantham, R. Nair, M. Tambe, and M. Yokoo. Winning Back the CUP for Distributed POMDPs: Planning Over Continuous Belief Spaces. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 2006.

[Web07] J.N. Webb. *Game Theory, Decisions, Interaction and Evolution*. Springer, 2007.

[WMH07] F. Weissel, Huber M.F., and U.D. Hanebeck. Test-Environment based on a Team of Miniature Walking Robots for Evaluation of Collaborative Control Methods. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, 2007.

[YCG+05] C. Yu, J. Chuang, B. Gerkey, G.J. Gordon, and A. Ng. Open-Loop Plans in Multi-Robot POMDPs. *Technical Report, Stanford CS Department*, 2005.

[YLB09] Shoham Y. and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[ZFS08] E. Zhou, M.C. Fu, and Marcus S.I. A Density Projection Approach to Dimension Reduction for Continuous State POMDPs. *Proceedings of the 47th IEEE Conference on Decision and Control*, 2008.