Stärkung deterministischer Strategien für POMDPs
Strengthening Deterministic Strategies for POMDPs

Leonore Winterer[1], Ralf Wimmer[2,1], Nils Jansen[3], Bernd Becker[1]

[1] Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau, Deutschland
{winterel, wimmer, becker}@informatik.uni-freiburg.de
[2] Concept Engineering GmbH, Freiburg im Breisgau, Deutschland,
[3]Radboud University Nijmegen, Nijmegen, Niederlande, n.jansen@science.ru.nl

Kurzfassung

Die Berechnung von Strategien für partiell beobachtbare Markov-Entscheidungsprozesse (POMDPs), die ein bestimmtes gewünschtes Verhalten garantieren, ist ein unentscheidbares Problem – optimale Strategien benötigen üblicherweise Kenntnis der gesamten Vorgeschichte, also unbegrenzten Speicher. Stattdessen stellen wir solver-basierte Algorithmen vor, um stationäre Strategien zu berechnen, die nur auf dem aktuellen Zustand anstatt der gesamten Vorgeschichte basieren. Da die Berechnung optimaler stationärer, aber randomisierter Strategien immer noch zu aufwändig ist, greifen wir auf deterministische stationäre Strategien zurück und nutzen Algorithmen für gemischt-ganzzahlige lineare Programme. Wir zeigen, wie ein gewisses Maß an Randomisierung erlaubt werden kann und wie eingeschränktes Wissen über die Vorgeschichte mithilfe von Zustands- und Observierungssplitting genutzt werden kann. Unsere Experimente belegen, dass diese Methoden kompetetiv mit traditionellen POMDP Algorithem sind.

Abstract

Computing strategies for partially observable Markov decision processes that ensure a certain desired behavior is an undecidable problem. Such optimal strategies typically require the knowledge of the full history, i. e., infinite memory. Instead, we propose solver-based algorithms for stationary algorithms that only depend on the current state instead of the whole history. Since computing optimal stationary, but randomized strategies is still too costly, we resort to deterministic stationary strategies and algorithms using mixed-integer programming. We show how to allow a certain amount of randomization and how to incorporate some knowledge of the history by applying state- and observation splitting. Our experiments demonstrate that these methods are competitive to traditional POMDP algorithms.

1 Einleitung

Partiell beobachtbare Markov Entscheidungsprozesse (POMDPs) sind ein probabilistisches Model, das den Nichtdeterminismus von Markov-Entscheidungsprozessen (MDPs) mit unvollständigen Informationen vereint. Das heißt, dass der aktuelle Zustand des Systems nicht in jedem Fall bekannt ist, sondern lediglich aus Beobachtungen erschlossen werden kann. Solche POMDPs können viele reale Anwendungsfälle simulieren, etwa immer dann, wenn ein Teil des Systems oder seiner Umgebung durch Sensoren nur ungenügend beobachtet werden kann – ein beliebtes Beispiel ist ein Roboter, der sich durch ein Labyrinth bewegt und seine aktuelle Position nur anhand der ihn umgebenden Wände herleiten kann. In diesem Fall kann dann etwa festgestellt werden, dass der Roboter an einer T-Kreuzung steht, bei mehreren solcher Kreuzungen im Labyrinth können diese allerdings nicht unterschieden werden.

Im Model Checking werden Strategien für POMDPs berechnet, die das Verhalten des Systems nach bestimmten Gesichtspunkten (etwa die Wahrscheinlichkeit, einen bestimmten Zustand zu erreichen, oder der durchschnittliche erwartete Reward, wenn bestimmte Aktionen im System eine Belohnung auslösen) optimieren oder versuchen, eine bestimmte Spezifikation (“ein schlechter Zustand wird in weniger als 0.1 % aller Ausführungen erreicht”) zu erfüllen. Da der Zustand eines POMDPs in der Regel nicht exakt bekannt ist, muss eine Strategie sich in Zuständen, die nicht unterschieden werden können, identisch verhalten. Allerdings können diese Strategien beliebig komplex werden, anstelle einer einzelnen Aktion eine Wahrscheinlichkeitsverteilung über die möglichen Aktionen liefern und auch die Vorgeschichte einer Beobachtung (also die bisherigen Beobachtungen und Aktionen) berücksichtigen. Im Allgemeinen ist das Problem, eine optimale Strategie in einem POMDP zu finden, unentscheidbar [1].

Eine Strategie, die nur die aktuelle Beobachtung zur Unterschiedung von Zuständen in Betracht zieht, nennt man *stationär*. Weist sie zudem jeder Beobachtung nur eine einzige Aktion (statt einer Wahrscheinlichkeitsverteilung) zu, spricht man von einer *deterministischen* Strategie. Stationäre und deterministische Strategien haben gegenüber allgemeinen Strategien viele Vorteile – sie können kompakt

abgebildet und somit auch auf Microcontrollern einfach implementiert werden. Außerdem ist die Berechnung optimaler deterministischer stationärer Strategien „nur“ NP-vollständig [2]. Aus diesen Gründen fokusieren wir uns in dieser Arbeit auf Methoden für die Berechnung optimaler stationärer Strategien für POMDPs. Wir modifizieren diese Methoden so, dass ein eingeschränktes Maß an Randomisierung und Vorgeschichte berücksichtigt werden kann. Dies führt zu starken Strategien, die zu den Ergebnissen traditioneller POMDP Algorithmen kompetetiv sind, ohne die Vorteile deterministischer Strategien aufzugeben.
Große Teile dieses Beitrags erschienen bereits in [5].

2 Strategien als Optimierungsprobleme

Um eine stationäre, aber randomisierte Strategie zu berechnen, die den erwarteten discounted Reward eines POMDPs maximiert, kann man das folgende einfache Optimierungsprogramm aufstellen. Die Variable $\sigma_{\lambda(s),\alpha}$ bezeichnet dabei die Wahrscheinlichkeit, mit in einem Zustand s mit der Beobachtung $\lambda(a)$ die Aktion α ausgeführt werden soll; v_s ist der erwartete discounted Reward, den man bei Verwendung dieser Strategie in Zustand s erhält, $0 < \beta < 1$ der Discount-Faktor.[1] Der Reward $v_{s_{\text{init}}}$ des Initialzustands s_{init} wird dabei maximiert gemäß der nachfolgenden Beschränkungen:

$$\text{maximize } v_{s_{\text{init}}} \tag{1a}$$

subject to:

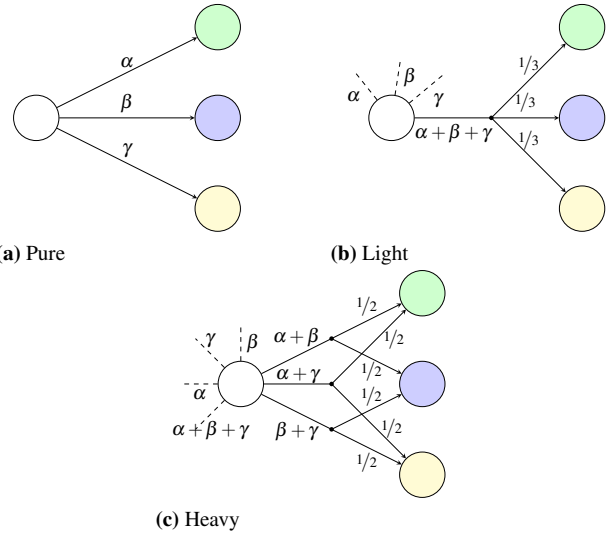
$$\forall s \in S\colon \quad \sum_{\alpha \in A(s)} \sigma_{\lambda(s),\alpha} = 1 \tag{1b}$$

$$\forall s \in S\colon \quad v_s \leq \sum_{\alpha \in A(s)} \sigma_{\lambda(s),\alpha} \big[r(s,\alpha) + \beta \cdot \sum_{s' \in \text{succ}(s,\alpha)} P(s,\alpha,s') \cdot v_{s'} \big] \tag{1c}$$

Bedingung (1b) stellt sicher, dass die berechnete Strategie gültig ist, dass sich also die Wahrscheinlichkeiten aller Aktionen pro Zustand zu 1 aufsummieren. Bedingung (1c) berechnet den Reward eines jeden Zustands auf rekursive Weise, in dem für alle Aktionen die Wahrscheinlichkeit, diese Aktion zu nehmen, mit den Wahrscheinlichkeiten und Rewards möglicher Nachfolgezustände multipliziert und zum Reward des aktuellen Zustands addiert wird. Das exakte Lösen des nicht-linearen Optierungsprogramms (1a)–(1c) nimmt sehr viel Rechenzeit und Speicher in Anspruch. Deshalb schlagen wir vor, die Berechnung auf deterministische Strategien zu beschränken, die statt einer Wahrscheinlichkeitsverteilung nur exakt eine Aktion pro Beobachtung ausgeben. Bei dem resultierenden gemischt-ganzzahligen linearen Programm (MILP) bleiben die Gleichungen (1a) und (1b) gleich. Statt der Gleichung (1c) erhalten wir

$$\forall s \in S\, \forall \alpha \in A(s)\colon \quad v_s \leq v^*_{\max} \cdot (1 - \sigma_{\lambda(s),\alpha}) + r(s,\alpha) + \beta \cdot \sum_{s' \in \text{succ}(s,\alpha)} P(s,\alpha,s') \cdot v_{s'}. \tag{1c'}$$

Strategien, die die Erreichbarkeitswahrscheinlichkeit oder die durchschnittliche Aufenthaltsdauer (Steady State Probability) in einer Menge von Zuständen maximieren, können analog berechnet werden. Hier muss allerdings zusätzlich darauf geachtet werden, dass die Wahrscheinlichkeiten nur für diejenigen Zustände berechnet wird, die unter der aktuell betrachteten Strategie vom Initialzustand aus erreicht werden können. Dazu können zusätzliche Bedingungen verwendet werden; in der Literatur wurden dafür Erreichbarkeits-Constraints [4] und Flussbedingungen [3] vorgeschlagen, die wir für unsere Zwecke anpassen und miteinander vergleichen.



(a) Pure **(b)** Light **(c)** Heavy

Abbildung 1 Ein POMDP ohne, mit “Light” und mit “Heavy” statischer Randomisierung

2.1 Statische Randomisierung

Für manche POMDPs kann die maximale Erreichbarkeitswahrscheinlichkeit bzw. der maximale Reward nicht ohne Randomisierung erreicht werden. Oft spielt dabei aber der genaue Wert der Verteilung keine Rolle, solange nur die Möglichkeit besteht, für eine Beobachtung zwei oder mehr verschiedene Aktionen zu wählen. Um dies auszunutzen, führen wir die sogenannte *statische Randomisierung* ein. Hierbei werden zusätzliche Aktionen eingeführt, die eine Gleichverteilung über (eine Teilmenge der) möglichen Aktionen abbilden. Wir unterscheiden im Folgenden zwischen „Pure“ (keine Randomisierung), „Light“ (eine neue Aktion, die über alle ursprünglichen Aktionen randomisiert) und „Heavy“ (eine neue Aktion pro nicht-leere Teilmenge der ursprünglichen Aktionen). Abbildung 1 zeigt diese drei Modi anhand eines Beispiels.

[1] Der Discount-Factor ist ein Maß dafür, um wieviel Belohnungen in der Zukunft weniger wert sind im Vergleich zu Belohnungen zum aktuellen Zeitpunkt. Er sorgt gleichzeitig dafür, dass der erwartete Reward in jedem Fall endlich ist.

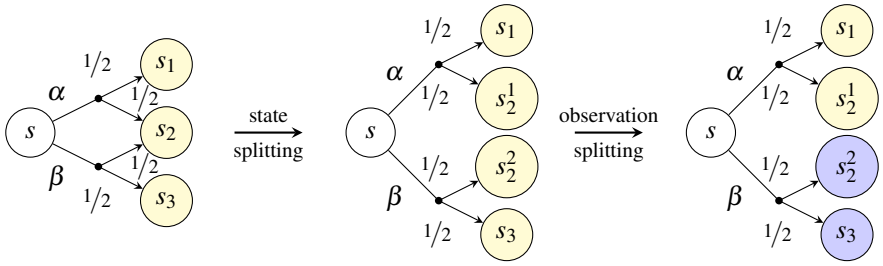


Abbildung 2 Aufteilung von Zuständen und Beobachtungen in einem POMDP. Die Beobachtung wird durch die Farbe der Zustände symbolisiert.

2.2 Aufteilung von Beobachtungen und Zuständen

Manchmal reicht auch Randomisierung nicht aus, um die für ein POMDP und seine Spezifikation optimale Strategie zu berechnen. In diesen Fällen wird üblicherweise die Vorgeschichte von Beobachtungen bei der Auswahl der nächsten Aktion berücksichtigt, um weitere Rückschlüsse darauf zu ziehen, in welchem Zustand man sich gerade befindet. Dies kann umgesetzt werden, indem man das Kreuzprodukt zwischen dem POMDP und einem endlichen Zustandsautomaten (FSC) berechnet, der eine vordefinierte Anzahl von Speicherzuständen abbildet. Allerdings führt dies zu einem starken Anwachsen des betrachteten Zustandsraums, was schon für wenige Speicherzustände problematisch werden kann. Hier machen wir uns zunutze, dass die Vorgeschichte oft nicht im gesamten POMDP, sondern nur in wenigen Zuständen wichtig wird, und führen nur dort punktuell zusätzliche Zustände ein.

In einem ersten Schritt stellen wir fest, dass mache Zustände mit gleicher Beobachtung alleine dadurch unterschieden werden können, dass man die Beobachtung des direkten Vorgängers sowie die zuletzt genutzte Aktion kennt. In diesem Fall teilen wir die Beobachtung auf und führen eine neue Beobachtung für die unterscheidbaren Zustände ein.

Nur wenige POMDPs ermöglichen es direkt, Beobachtungen auf diese Art und Weise aufzuteilen. In diesen Fällen können wir versuchen, zunächst Zustände anhand ihrer vorangehenden Beobachtungen und Aktionen aufzuteilen, um anschließend die Zuweisung verschiedener Beobachtungen zu ermöglichen. Abbildung 2 zeigt dieses Vorgehen im Bild – auf der linken Seite kann die gelbe Beobachtung nicht direkt aufgeteilt werden, da der Zustand s_2 sowohl mit der Aktion α als auch β erreicht werden kann. Deshalb spalten wir zunächst s_2 in zwei Kopien s_2^1 und s_2^2 auf. In einem vollständigen POMDP würden beide Zustände die gleichen ausgehenden Übergange erhalten, so dass das Verhalten des Gesamtsystems nicht beeinträchtigt wird. Nun kann – da s_1 und s_2^1 nur mittels der Aktion α und s_2 sowie s_2^2 nur mittels β erreicht werden können – die gelbe Beobachtung aufgeteilt und eine neue Beobachtung für die unteren beiden Zustände eingeführt werden.

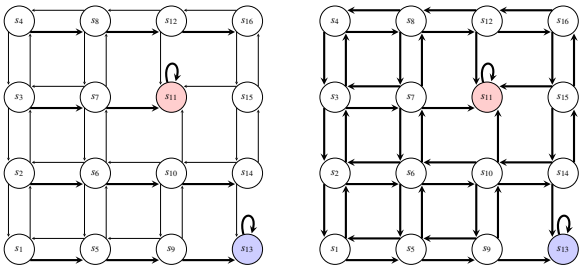
Eine Schwierigkeit bleibt es, herauszufinden, welche Zustände geteilt werden müssen, um die Qualität der berechneten Strategie zu verbessern. Dies ist meist nicht auf den ersten Blick ersichtlich, und ein übereifriges Aufteilen von Zuständen kann wiederum zu einem exponentiellen Aufblähen des Systems führen. Für diesen Zweck stellen wir eine Heuristik vor, die in einer Schleife immer wieder einzelne Kandidaten für eine Aufteilung ermittelt und so neue, bessere Strategien berechnet.

3 Experimente

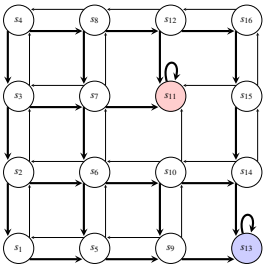
Wir testen unsere Methode für eine Anzahl von Benchmarks aus verschiedenen Quellen. Das hier vorgestellte Beispiel *4×4grid_avoid* stammt von der Seite des PRISM-pomdp model checker[2], der, genau wie unser Ansatz, approximative Strategien für die Maximierung der Erreichbarkeitswahrscheinlichkeit berechnen kann.

Abbildung 3 zeigt, welche Strategien unser Ansatz hier für verschiedene Randomisierungsmodi berechnet, um die Wahrscheinlichkeit zu maximieren, den blauen s_{13} Zustand zu erreichen, ohne im roten s_{11} Zustand steckenzubleiben. Alle weißen Zustände teilen sich eine Beobachtung und müssen in einer deterministischen Strategie deshalb die selbe Aktion auswählen. Als Startzustand wird zufällig einer der weißen Zustände ausgewählt. Die **fettgedruckten** Aktionen wurden von der aktuellen Strategie mit positiver Wahrscheinlichkeit ausgeführt. Deutlich sieht man die Steigerung der Erreichbarkeitswahrscheinlichkeit von 0.21 beim *Pure* Ansatz (nur die Zustände s_1, s_5 und s_9 können den Zielzustand erreichen) zu 0.85 bei Anwendung von *Heavy* statischer Randomisierung (alle Zustände außer s_{11} erreichen den Zielzustand mit positiver Wahrscheinlichkeit) bei fast gleichbleibender Laufzeit. Kommt zusätzlich unsere Heuristik zu Zustandsaufteilungen zum Einsatz, kann die Wahrscheinlichkeit bei einer Laufzeit von 9.3s auf 0.93 gesteigert werden. PRISM-pomdp errechnet eine maximale Wahrscheinlichkeit von 0.96 innerhalb von 346.9 Sekunden – eine vergleichsweise geringe Verbesserung der Wahrscheinlichkeit bei einem Vielfachen der Laufzeit.

[2] http://www.prismmodelchecker.org/files/rts-poptas/



(a) Pure – 0.21 – 0.1s **(b)** Light – 0.3 – 0.2s



(c) Heavy – 0.85 – 0.1s

Abbildung 3 Strategien, die für den 4×4grid_avoid Benchmark mithilfe unterschiedlicher Randomisierungs-Modi berechnet wurden.

4 Ausblick

Neben der Berechnung handlicher, aber effizienter Strategien für POMDPs ermöglicht es unser Ansatz auch, beliebig viele Spezifikationen miteinander zu verbinden. So kann etwa die Wahrscheinlichkeit maximiert werden, einen Zielzustand zu erreichen, während gleichzeitig die Anzahl der benötigten Schritte unter einer bestimmten Schranke bleibt, oder der regelmäßige Aufenthalt in einer Menge von Zuständen garantiert wird. Außerdem ermöglichen die (fast) deterministischen Strategien das einfache Aufspüren von Fehlern im System, die ausgebessert werden müssen, um eine Erfüllung der Spezifikation zu garantieren.

5 Literatur

[1] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In Jim Hendler and Devika Subramanian, editors, *Proc. of the 16th National Conf. on Artificial Intelligence (AAAI)*, pages 541–548, Orlando, FL, USA, 1999. AAAI Press.

[2] Martin Mundhenk. *The Complexity of Planning mit Partially-Observable Markov Decision Processes*. Habilitationsschrift, Friedrich-Schiller-Universität Jena, Jena, Germany, 2000.

[3] Alvaro Velasquez. Steady-state policy synthesis for verifiable control. In Sarit Kraus, editor, *Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence (IJCAI)*, pages 5653–5661. ijcai.org, 2019.

[4] Ralf Wimmer, Nils Jansen, Erika Ábrahám, Joost-Pieter Katoen, and Bernd Becker. Minimal counterexamples for linear-time probabilistic verification. *Theoretical Computer Science*, 549:61–100, 2014.

[5] Leonore Winterer, Ralf Wimmer, Nils Jansen, and Bernd Becker. Strengthening deterministic policies for POMDPs. In Ritchie Lee, Susmit Jha, and Anastasia Mavridou, editors, *Proc. of the 12th NASA Formal Methods Symposium (NFM)*, volume 12229 of *Lecture Notes in Computer Science*, pages 115–132, NASA Ames Research Center, Moffett Field, CA, USA, 2020. Springer.