Frankfurt University of Applied Sciences
Fachbereich 2: Informatik und Ingenieurwissenschaften

Grundlagen adaptiver Wissenssysteme (SS2024)

Prof. Dr. Thomas Gabel

Projektvorschläge

Vorbemerkungen

Diese Zusammenstellung fasst für Sie alle Vorschläge für ein von Ihnen im Laufe des Semesters umzusetzendes Projekt zusammen, in dessen Rahmen Sie einen der in der Vorlesung besprochenen Algorithmen implementieren und praktisch anwenden werden. Außerdem werden Sie ihr Projekt mit einer Live-Demo und kurzen Präsentation im Rahmen der mündlichen Prüfung zum Ende des Semesters präsentieren. Die Projekte werden nach einer FCFS-Strategie[1] mittels eine campUAS-Abstimmung vergeben und sollten i.d.R. von Ihnen in Teams mit zwei Personen bearbeitet werden. Zusätzlich zu den von mir vorgeschlagenen Projekten dürfen Sie Ihrerseits auch gern eigenen Projektideen vorschlagen und *nach Rücksprache* mit mir als "Ihr Projekt" verwenden. Bitte beachten Sie, dass Sie sich bis spätestens vier Wochen vor Semesterende für ein Projektthema entschieden haben müssen.

PROJEKTVORSCHLAG 1: Optimales Verhalten in der Klausur

Schwerpunktthema: Rückwärts-DP

Personenanzahl: 1

Mindestvoraussetzung: VL-Kapitel 4

Implementieren Sie den Backward-DP-Algorithmus und ermitteln Sie die optimale Strategie für die in Aufgabe 1 des Übungsblattes 1 geschilderte Problemstellung. Was ist der Wert von V^{π^*}?

Erweitern Sie Ihre Implementierung und gestalten Sie sie so generisch, dass diese auch die optimale Strategie ermitteln kann für eine Klausur mit $N > 5$ Aufgabenlösungsversuchen, für Klausuren mit einer variierenden Anzahl von Aufgaben sowie mit variabel spezifizierbaren Punkten und Aufgabenlösungswahrscheinlichkeiten!

PROJEKTVORSCHLAG 2: Apres Ski in Südtirol

Schwerpunktthema: Rückwärts-DP

[1] First Come First Serve: Wer sich als erstes meldet, bekommt den Zuschlag.

Personenanzahl: 1-2

Mindestvoraussetzung: VL-Kapitel 4

Für die kommende Wintersaison (4 Monate, d.h. Dezember bis März) wurden Sie als Pächter einer Apres-Ski-Hütte in den Südtiroler Alpen eingestellt. Ihre Aufgabe soll es sein, im Laufe der Saison durch Verkauf von Getränken und Nahrungsmitteln so viel Profit wie möglich zu machen. Sie können wie folgt für Nachschub auf der Hütte sorgen:

- Morgens um 7 Uhr können Sie sich per Seilbahn oder mit dem Hubschrauber mit Getränken/Nahrungsmitteln versorgen lassen.
- Nachmittags um 14 Uhr können Sie sich mit dem Hubschrauber mit Getränken/Nahrungsmitteln versorgen lassen.
- Per Seilbahn können Sie bis zu 3 Einheiten Getränke/Nahrungsmittel transportieren. Die Seilbahnfahrt verursacht Kosten von 10€ pro transporterter Einheit Getränk/Nahrungsmittel.
- Per Hubschrauber können Sie bis zu 4 Einheiten Getränke/Nahrungsmittel transportieren. Der Hubschrauberflug verursacht Fixkosten in Höhe von 50€ sowie Kosten von 5€ pro transporterter Einheit Getränk/Nahrungsmittel.

Zu bedenken ist, dass jede Einheit[2] eines Getränks im Einkauf für Sie Kosten von 20€ verursacht, während der Verkauf einer Einheit Getränke einen Erlös von 80€ erbringt. Bei den Nahrungsmitteln kostet jede Einheit[3] im Einkauf 25€ und bringt einen Verkauferlös von 150€.

Ihr Lagerplatz in der Skihütte ist begrenzt. Sie können bis zu 10 Einheiten Getränke lagern, während ihr Tiefkühlfach Lagerplatz für bis zu 20 Einheiten Nahrungsmittel bietet. Am Beginn der Saison starten Sie mit leerem Lager. Sollten zum Ende der Saison noch Waren in Ihrer Skihütte lagern, so können Sie diese nicht mehr weiterverkaufen und auch nicht selbst behalten; für Sie ergeben sich also Terminalkosten.

Ihr individuelles Verkaufs- und Marketingtalent modellieren wir vereinfachend, indem wir annehmen, dass jeder Skifahrer, der bei Ihrer Skihütte vorbeikommt, mit folgenden Wahrscheinlichkeiten ein Getränk erwirbt bzw. eine Mahlzeit einnimt (hierbei unterscheiden wir zwischen Vormittag/Mittag (bis 14 Uhr, inkl. Mittagessenszeit) sowie Nachmittag (bis Schließung der Lifte, d.h. inkl. Apres-Ski-Zeit):

- Vormittag und Mittag
  - Wahrscheinlichkeit, ein Getränk zu erwerben: $p_{getr}^{vorm} = 0.3$
  - Wahrscheinlichkeit, eine Mahlzeit zu verzehren: $p_{essen}^{vorm} = 0.2$
- Nachmittag und Apres Ski

[2] Eine Einheit Getränke umfasst 20 Getränke (z.B. eine Kiste Bier).

[3] Als Einheit sehen wir hier vereinfachend 5 Mahlzeiten an, z.B. eine Packung mit fünf Schnitzeln und ausreichend Pommes.

Anzahl Besucher	Wkt. Vormittag	Wkt. Nachmittag
0	0.1	0.0
25	0.1	0.05
50	0.2	0.05
75	0.3	0.3
100	0.2	0.3
125	0.1	0.2
150	0.0	0.1

Tabelle 1: Apres-Ski-Hütten-Besucher pro Tag

- Wahrscheinlichkeit, ein Getränk zu erwerben: $p_{getr}^{nachm} = 0.7$
- Wahrscheinlichkeit, eine Mahlzeit zu verzehren: $p_{essen}^{nachm} = 0.1$

Für die Anzahl der Skifahrer, an Ihrer Hütte vorbeikommen, nehmen wir eine Verteilung gemäß Tabelle 1 an.

Jeden Tag ergeben sich daher zwei Entscheidungszeitpunkte, bei denen unterschiedliche Aktionen zur Verfügung stehen. Achtung: Sollten Skifahrer bei Ihner Hütte vorbeikommen und ein Getränk/Nahrungsmittel erwerben wollen, während Ihr lokales Lager leer ist, so werden diese Kunden nicht bedient (und Ihnen entgehen diese Einnahmen).

Implementieren Sie den Backward-DP-Algorithmus und ermitteln Sie die optimale Strategie für Sie als Apres-Skihütten-Pächter. Was ist der Wert von V^{π^*}? Finden Sie eine anschauliche Darstellung für die ermittelte Strategie.

PROJEKTVORSCHLAG 3:
Richtiges Verhalten bei der Parkplatzsuche

Schwerpunktthema: Wertiterationsverfahren

Personenanzahl: 1-2

Mindestvoraussetzung: VL-Kapitel 5

Ein lernfähiger Agent ist auf dem Weg zum Kino und befindet sich auf der Suche nach einem Parkplatz. In der zum Kino führenden Einbahnstraße, durch die er gerade fährt, befinden sich 100 Parkplätze (siehe Abbildung 1), auf denen kostenfrei geparkt werden darf.

Bei jedem Parkplatz, an dem der Agent entlangfährt, hat er die Möglichkeit, diesen zu nehmen (sofern frei) oder aber weiter zu fahren. Die Parkplätze sind durchnummeriert, der Agent startet beim Parkplatz 100. Unmittelbar nach dem letzten Parkplatz (Platz 1) folgen sowohl das Kino, als auch ein großes Parkhaus, in dem man aber zum Parken bezahlen muss.

Aus Sicht des Agenten fallen Kosten an, die von der Nummer i des gewählten Parkplatzes abhängen ($c = i$) und somit reflektieren, dass der Agent von Parkplätzen mit

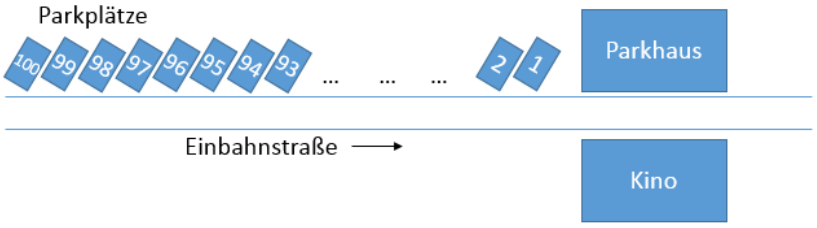

Abbildung 1: Das Parkplatz-Dilemma

größeren Nummern einen längeren Fußweg bis zum Kino in Kauf nehmen muss. Ein Rückwärtsfahren ist für den Agenten ausgeschlossen, da es sich um eine Einbahnstraße handelt. Wenn der Agent beim Parkplatz 1 angenkommen ist und dieser belegt sein sollte, so bleibt im nur die Möglichkeit, im teuren Parkhaus zu parken. Dies soll mit hohen Terminalkosten von $c = 200$ modelliert werden.

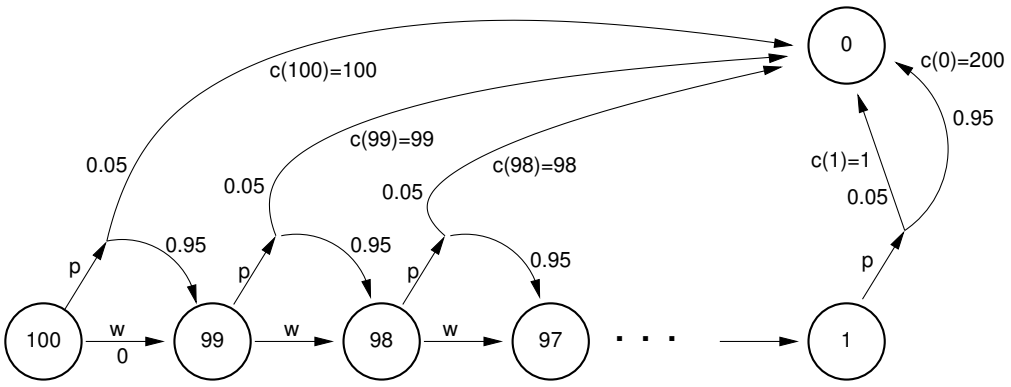

Abbildung 2: Zustände im Parkplatzproblem

Die Wahrscheinlichkeit, dass ein beliebiger Parkplatz frei ist wird, beträgt $p = 0.05$. Dementsprechend ist ein jeder Parkplatz mit der Gegenwahrscheinlichkeit von $1 - p$ bereits besetzt. In Abbildung 2 sind die Gegebenheiten der MDP-Umgebung des Parkplatzproblems grafisch zusammengefasst.

(a) Geben Sie eine vollständige Spezifikation des Problems als Markov'scher Entscheidungsprozess an.

(b) Implementieren Sie den Wertiterationsalgorithmus (Value Iteration) für das gegebene Problem und beantworten Sie so die Frage: "Wann sollte man parken?"

(c) Variieren Sie die Parameter der Problemstellung. Wiederholen Sie ihre Experimente für reduzierte Terminalkosten von $c = 100$ sowie für eine höhere Wahrscheinlichkeit ($p = 0.2$), mit der ein Parkplatz frei ist.

PROJEKTVORSCHLAG 4:
Wann soll ich kaufen?

Schwerpunktthema: Rückwärts-DP

Personenanzahl: 2

Mindestvoraussetzung: VL-Kapitel 4

Sie wollen einem geliebten Menschen zu einem besonderen Anlass a ein größeres Geschenk g machen. In einem marktwirtschaftlichen Wirtschaftssystem unterliegt der Preis des Geschenks $\mathcal{P}(g)$ regelmäßigen Schwankungen. Selbstverständlich wollen Sie sich auch nicht "übers Ohr hauen lassen" und das Geschenk zu einem möglichst günstigen Preis erwerben.

Folgende Rahmenbedingungen charakterisieren die Aufgabenstellung:

- Der Preis des Geschenks ändert sich stündlich. Sie bzw. der für Sie arbeitende Agent kann stündlich die Entscheidung treffen, ob er zum aktuellen Preis das Geschenk kaufen möchte, oder ob er weiter warten will.
- Da das Geschenk zu einem bestimmten Zeitpunkt vorliegen soll, z.B. weil es als Geburtstagsgeschenk rechtzeitig gebraucht wird, gibt es einen spätestmöglichen Zeitpunkt t_{final}, zu dem der Agent das Produkt gekauft haben sollte. D.h. ein weiteres Warten (auf einen noch günstigeren Preis) ist dann nicht mehr möglich.
- Wir nehmen an, dass es für das Geschenk einen hypothetischen Minimalpreis $\mathcal{P}_{min} > 0$ und einen Maximalpreis $\mathcal{P}_{max} > \mathcal{P}_{min}$ gibt, die nicht unter- bzw. überschritten werden können.
- Den Preis des Geschenks zu Beginn des Beobachtungszeitraum ($t = 0$) bezeichnen wir mit $\mathcal{P}_0(g) \in [\mathcal{P}_{min}, \mathcal{P}_{max}]$.
- Aufgrund der auf Angebot und Nachfrage basierenden Gesetze des Marktes nehmen wir an, dass der Preis des Geschenks sich von Stunde zu Stunde wie folgt ändert:

$$\mathcal{P}_{t+1}(g) = max(\mathcal{P}_{min}, min(\mathcal{P}_{max}, \mathcal{P}_t(g) + k\delta))$$

wobei sich der Wert k stochastisch wie folgt ergibt:

$$k = \begin{cases} -1 & \text{mit Wahrscheinlichkeit } \begin{cases} \frac{\varrho}{2} & \text{wenn } \varrho \leq 1 \\ \frac{1}{2\varrho} & \text{sonst} \end{cases} \\ 1 & \text{sonst} \end{cases}$$

wobei $\varrho = \frac{\mathcal{P}_t(g) - \mathcal{P}_{min}}{\mathcal{P}_{max} - \mathcal{P}_t(g)}$. Hierbei steht δ für die absolute, mögliche Preisschwankung in einer Stunde; d.h. der Preis sinkt oder steigt um den Betrag δ pro Stunde.

- Ziel des Agenten ist es, das Geschenk zum bestmöglichen (geringsten) Preis zu kaufen. Kosten entstehen aus Sicht des Agenten also ausschließlich durch den Kaufpreis im Moment des Kaufes. Falls das Geschenk nicht bis zur Frist t_{final} gekauft worden ist, so setzen wir zudem Terminalkosten von $5 \cdot \mathcal{P}_0(g)$ an, um das moralische Übel zu modellieren, dass Sie dem geliebten Menschen zum fraglichen Zeitpunkt kein Geschenk überreichen können.

(a) Geben Sie eine vollständige Spezifikation des Problems als Markov'scher Entscheidungsprozess an.

(b) Implementieren Sie den Rückwärts-DP-Algorithmus (Backward Dynamic Programming) für das gegebene Problem und beantworten Sie so die Frage "Wann sollte der Agent das Geschenk kaufen?".
Verwenden Sie hierbei folgende Parametereinstellungen: $t_{final} = 500$ (ca. 3 Wochen), $\delta = 10$ (Preisschwankung von 10 Euro pro Stunde), $\mathcal{P}_0(g) = 100$ (Startpreis von 100 Euro), Minimal- und Maximalpreis von $\mathcal{P}_{min} = 0$ bzw. $\mathcal{P}_{max} = 200$ Euro.

(c) Finden Sie eine anschauliche Darstellung Ihrer Ergebnisse (sowohl für Wertfunktionen als auch für die Strategie).

(d) Variieren Sie die Rahmenparameter (t_{final}, δ, $\mathcal{P}_0$, $\mathcal{P}_{min}$, $\mathcal{P}_{max}$) und gelangen Sie so zu allgemeineren Aussagen über günstige Kaufzeitpunkte.

(e) Führen Sie eine Recherche in der wirtschaftswissenschaftlichen Literatur durch, finden Sie dort Ansätze, nach denen der Preisverlauf üblicherweise modelliert wird (also anders, als mittels der durch ϱ definierten Wahrscheinlichkeiten), und integrieren Sie diese in Ihre Lösung.

PROJEKTVORSCHLAG 5:
Optimale Strategie für das Leben

Schwerpunktthema: Wertiterationsverfahren

Personenanzahl: 2

Mindestvoraussetzung: VL-Kapitel 5

Das Leben eines lernfähigen Agenten sei mit dem MDP modelliert, der in Abbildung 3 dargestellt ist. Hierbei handelt es sich um eine verfeinerte Version des Leben-MDP, der bereits aus der Vorlesung bekannt ist.

Folgende Besonderheiten sind bei der hier gegebenen Modellierung zu beachten:

- Es wird eine Modellierung auf Basis von Belohnungen (anstatt Kosten) gewählt, so dass es Ziel des Agenten ist, die Summe der zu erwartenden Belohnungen zu maximieren. Der Agenten erhält beispielsweise eine Belohnung von 1, wenn er im Zustand $INIT\_MENSCH$ ist und hier die Aktion $FAULENZE$ ausführt.

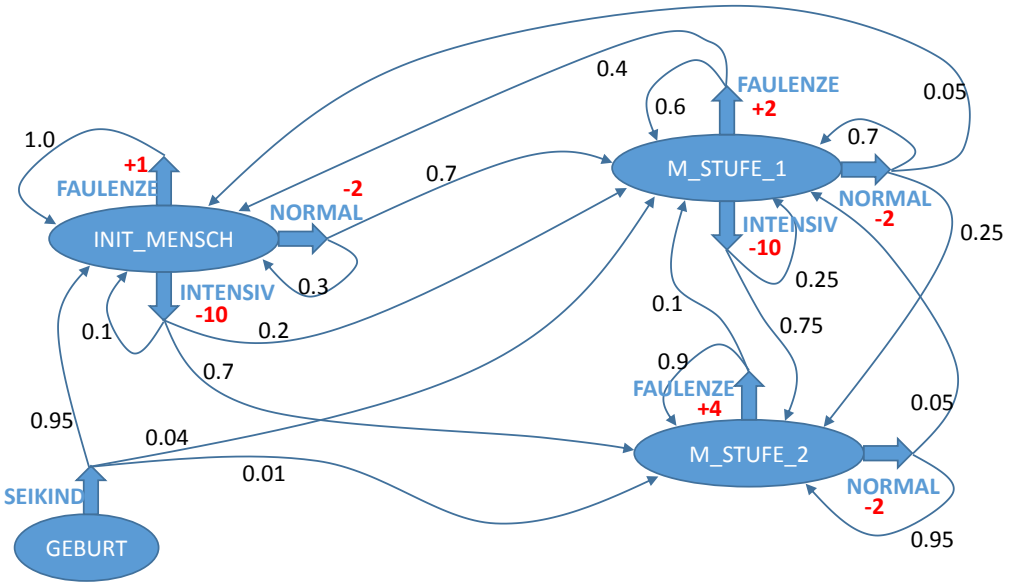

Abbildung 3: Lebens-MDP

- Die Aktionsmengen, die dem Agent zur Verfügung stehen, sind zustandsabhängig. D.h. dass die Menge A der verfügbaren Aktionen unterschiedlich viele Elemente enthält, je nach dem, in welchem Zustand er sich befindet. Es ist daher zielführend, nicht mit einer fixen Menge A zu arbeiten, sondern zustandsabhängige Aktionsmengen $A(s)$ zu definieren und auf diesen zu operieren. Welche Aktionen in welchem Zustand verfügbar sind, ergibt sich ebenfalls aus der Abbildung.

Der Agent kann nichts tun (sich entspannen, faulenzen, Aktion FAULENZE) oder arbeiten bzw. sich weiterbilden. Bei letzterem kann er wählen zwischen normaler Arbeitsintensität (NORMAL) und hoher Arbeitsintensität (INTENSIV). Aktionen sind in der Abbildung in blauer Schrift gekennzeichnet, zugehörige Belohnungen in rot und die Übergangswahrscheinlichkeiten in schwarz. Der Zustand $INIT\_MENSCH$ korrespondiert zur "Jugend" des Agenten, in der er sich zum Nichtstun, zu einem Bummelstudium oder zu einem Extremstudium entscheiden kann. Die Stufen 1 und 2 korrespondieren zum Arbeitsleben, wo der Agent ebenfalls mit lockerer Einstellung (NORMAL) oder mit hoher Zielstrebigkeit (INTENSIV) arbeiten kann.

Ihre Aufgabenstellung umfasst die folgenden Teile:

(a) Geben Sie eine vollständige formale Spezifikation (d.h. in mathematischer Notation) des Problems als Markov'scher Entscheidungsprozess an.

(b) Implementieren Sie den Wertiterationsalgorithmus (Value Iteration) für das gegebene Problem und ermitteln Sie so die optimale Strategie für den Agenten, wobei Sie einen Diskontierungsfaktor von $\gamma = 0.99$ zugrunde legen.

(c) Ändern Sie die Weitsichtigkeit des Agenten. Variieren Sie hierzu den Diskontierungsfaktor und ermitteln Sie so, welche Änderungen an der optimalen Strategie sich in Abhängigkeit der geänderten Diskontierung ergeben. Stellen Sie Ihre Resultate anschaulich und übersichtlich dar.

PROJEKTVORSCHLAG 6:
Dynamisches Programmieren bei "Schlag den Star"

Schwerpunktthema: Wertiterationsverfahren

Personenanzahl: 2

Mindestvoraussetzung: VL-Kapitel 5

Betrachten Sie die volle Problemstellung des Spiels "Würfeln" (vgl. Aufgabe aus den Übungen), allerdings nehmen Sie davon Abstand, das Problem als Multi-Agenten-Problem zu untersuchen, und lassen daher die Existenz eines Gegenspielers außer Acht. Zielstellung des Spielers ist es also, möglichst schnell (d.h. möglichst wenige Male den Würfel abzugeben) 50 Punkte erwürfelt und auf seinem Konto gutgeschrieben zu haben.

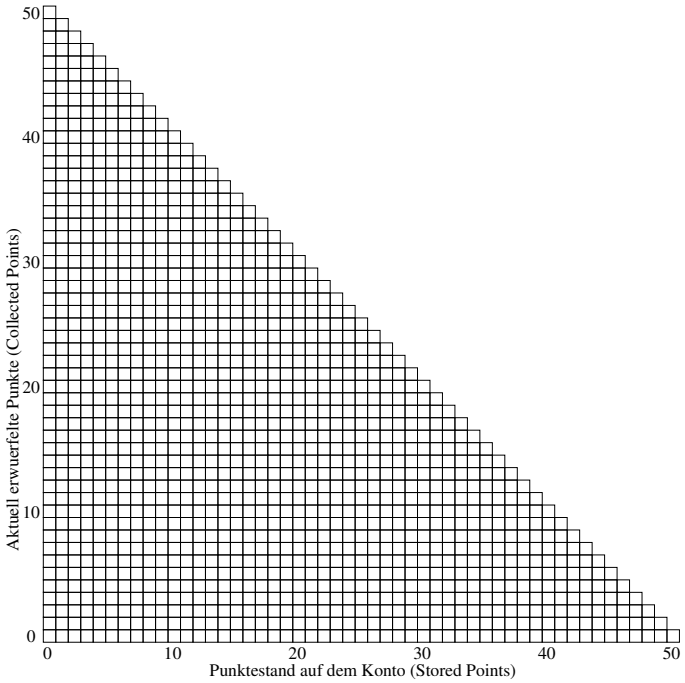

Abbildung 4: Strategieformular für das Würfel-Spiel

(a) Reflektieren Sie über das Problem und entwickeln Sie eine Vorstellung vom Aussehen der optimalen Strategie. Tragen Sie ihre intuitive Schätzung in Abbildung 4 ein, indem Sie Zustände, in denen Sie die Aktion "WÜRFEL" wählen würden, schwärzen,

und Zustände, in denen Sie die Aktion "GIBAB" wählen würden, unausgefüllt belassen.

(b) Modellieren Sie die beschriebene Version des Spieles als SKP-Problem. Begründen Sie Ihre Definition der direkten Kosten.

(c) Implementieren Sie den Wertiterationsalgorithmus für das Problem und ermitteln Sie so die optimale Strategie.

(d) Zeichnen Sie die von Ihnen ermittelte optimale Strategie auch in das "Strategieformular" (Abbildung 4) ein. Welche Bereiche des Zustandsraumes können unter dieser Strategie überhaupt erreicht werden?

PROJEKTVORSCHLAG 7:
Strategieoptimierung in der Autovermietung

Schwerpunktthema: Strategieiterationsverfahren

Personenanzahl: 2

Mindestvoraussetzung: VL-Kapitel 6

Im Auftrag der großen europäischen Autovermietung GEAV mit deutschem Hauptsitz in Berlin verwalten Sie zwei lokale Niederlassungen (in Frankfurt und Offenbach). Tagtäglich kommen Leute zu Ihnen, um Autos zu mieten. Für jeden vermieteten Wagen zahlt Ihnen die GEAV 20€. Wenn Sie an demjenigen Standort, an dem ein Kunde zu Ihnen kommt (also in Frankfurt oder in Offenbach), gerade kein Auto zur Verfügung haben, so entgeht Ihnen dieses Geschäft.

Für eine Neuvermietung stehen Autos grundsätzlich am Folgetag des Tages zur Verfügung, an dem sie zurückgegeben wurden. Um sicherzugehen, dass Sie am nächsten Tag an jedem Ihrer beiden Standorte ausreichend viele PKWs zur Verfügung haben, können Sie über Nacht eine beliebige Anzahl von Autos zwischen Ihren beiden Frankfurt und Offenbach verschieben. Durch Benzinverbrauch und Verschleiß verursacht Ihnen diese Aktion Kosten in Höhe von 4€ pro verschobenem Wagen.

Die Anzahl pro Tag an jedem Standort eintreffender Neukunden (zur Anmietung eines Wagens) sowie die Anzahl zurückgegebener Wagen an jedem Standort ist Poissonverteilt. Die Wahrscheinlichkeit p, dass an einem Standort an einem spezifischen Tag also genau n Wagen ausgeliehen/zurückgegeben werden, ergibt sich somit gemäß $p = \frac{\lambda^n}{n!} e^{-\lambda}$. Die Werte des Parameters λ für Ausleihe/Rückgabe sowie Standort Frankfurt/Offenbach sind in Tabelle 2 gegeben.

Zur Vereinfachung des Problems sei angenommen, dass sowohl in Frankfurt als auch in Offenbach maximal 20 Parkplätze vor Ihrer Filiale zur Verfügung stehen. Sollte der Fall eintreten, dass an einem Tag an einem Standort so viele Wagen zurückgegeben werden, dass der Platz nicht ausreicht, so werden die überschüssigen Autos zur Hauptzentrale zurücktransferiert, ohne dass für Sie dabei Kosten entstehen; diese Wagen verschwinden also aus der Problemstellung. Ferner nehmen wir an, dass Sie aufgrund eines begrenzten

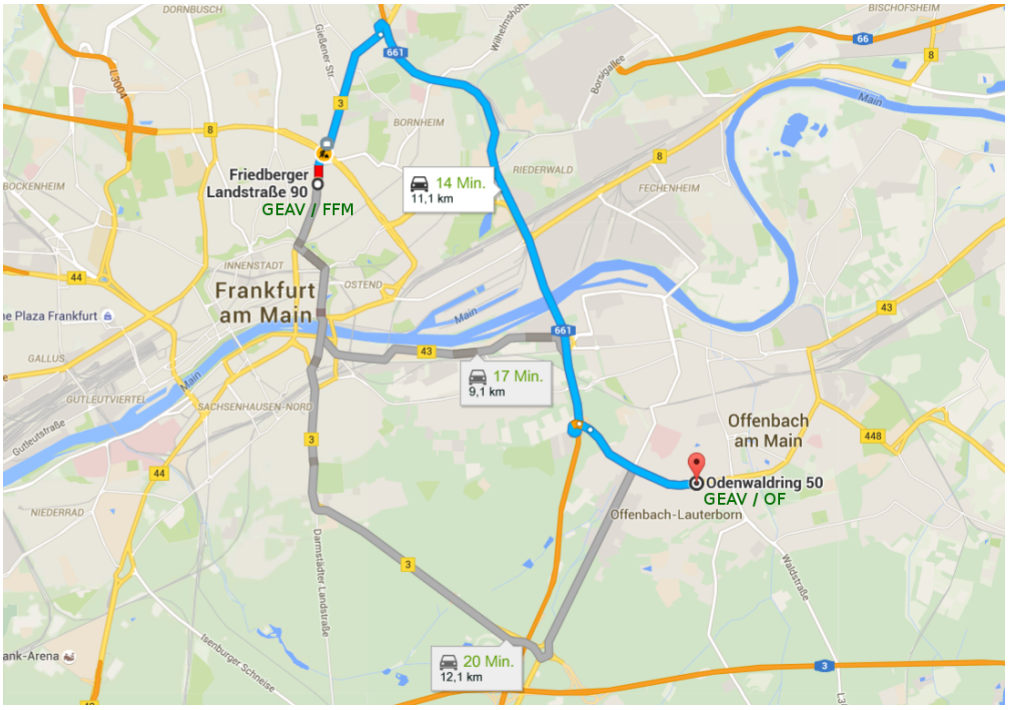

Abbildung 5: Niederlassungen der GEAV im Rhein-Main-Gebiet

	Standort Frankfurt	Standort Offenbach
Ausleihe	3	4
Rückgabe	3	2

Tabelle 2: Parameter λ der Poisson-Verteilung für die Autovermietung

Mitarbeiterstabs pro Tag nur maximal 5 PKWs zwischen Frankfurt und Offenbach (und umgekehrt) transferieren können.

Modellieren Sie das Problem als MDP mit unendlichem Horizont und einem Diskontierungsfaktor von $\gamma = 0.9$, wobei ein Zeitschritt zu einem Tag korrespondieren soll. Implementieren Sie den Strategieiterationsalgorithmus und wenden Sie Ihn auf die skizzierte Problemstellung an. Ermitteln Sie so die optimale Strategie π^*. Nach wie vielen Iterationen des Strategieiterationsverfahrens wird die optimale Strategie gefunden? Finden Sie eine ansprechende Visualisierung, um die gefundene optimale Strategie sowie deren Entwicklung über die einzelnen Iterationsschritte darzustellen.

PROJEKTVORSCHLAG 8:
X-Wing Fighter Control

Schwerpunktthema: generalisierte Strategieiteration mit MC-Rollouts

Personenanzahl: 2

Mindestvoraussetzung: VL-Kapitel 8

Die Macht ist mit Ihnen. Ihre Aufgabe besteht darin, Ihren Angriffsjäger durch einen schmalen Schacht zu navigieren, an dessen Ende Sie den entscheidenden Phaser-Schuss zur Zerstörung des Todessterns abgeben können. Im Rahmen dieses Projekts werden zwei Topologien der zu durchfliegenden Schächte des Todessterns untersucht, die in Abbildung 6 dargestellt sind.

Zielstellung für ihren lernfähigen Agenten muss es sein, den Schacht so schnell wie möglich zu durchfliegen und zur Austrittsstelle zu gelangen. Selbstverständlich sollen dabei Berührungen mit den Schachtwänden vermieden werden. Der Flieger befindet sich zu jedem Zeitpunkt in einer diskreten Zelle des in Abbildung 6 dargestellten Gitters; seine Geschwindigkeit ist ebenfalls diskret und gibt an, um wie viele Gitterzellen der Flieger pro Zeitschritt in x- sowie in y-Richtung weiterbewegt wird. Die zur Verfügung stehenden Aktionskomponenten sind *Beschleunigen* (B), *Geschwindigkeit halten* (H) und *Verlangsamen* (V). Hierbei gilt, dass jede Aktion sich durch eine Aktionskomponenten in x- sowie in y-Richtung auszeichnet und die Geschwindigkeit in die jeweilige Richtung um eine Einheit erhöht (B), konstant lässt (H) oder aber um eine Einheit reduziert (B). Die Menge aller Aktionen umfässt somit 9 Elemente.

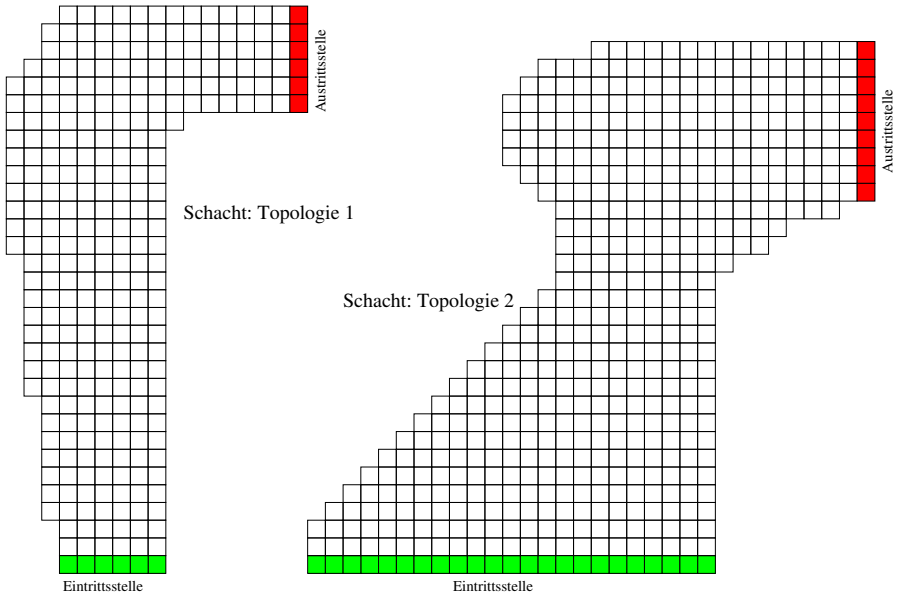

Abbildung 6: Topologien (Versionen 1 und 2) des zu durchfliegenden Schachtes

Zur Vereinfachung ist der Wertebereich möglicher Geschwindigkeiten auf positive Werte und Wert kleiner fünf eingeschränkt (also $[0, 4]$). Außerdem ist es (mit Ausnahme des Startzustandes) untersagt, die Geschwindigkeit in beide Richtungen gleichzeitig auf 0 zu reduzieren. Eine Episode beginnt in einem zufälligen gewählten Eintrittspunkt auf der Eintrittslinie und endet beim Überqueren der Austrittstelle. In jedem Zeitschritt, in dem der Agent im Schacht bleibt, erfährt er Kosten von 1, in jedem Zeitschritt, in

dem er die Wand berührt, Kosten von 5. Berührungen mit der Wand resultieren nicht in einer Zerstörung der Raumschiffs, sondern darin, dass das Schiff nur um eine Gitterzelle entlang der Wand weiterbewegt wird (horizontal und/oder vertikal). Unter diesen Voraussetzungen ist sichergestellt, dass alle denkbaren Strategien stets die Zielllinie erreichen.

Um die Aufgabenstellung etwas interessanter zu gestalten, nehmen wir an, dass in 50% der Zeitschritte die Bewegung entweder in horizontaler oder in vertikaler Richtung um einen Schritt weiter ausgeführt wird, als die aktuelle Geschwindigkeit es eigentlich vorschreiben würde.

Berechnen Sie für jeden möglichen Startzustand, also für jeden möglichen Eintrittspunkt auf der Eintrittslinie die optimale Strategie für Ihren X-Wing Fighter. Verwenden Sie hierzu den Algorithmus zur Monte-Carlo-Strategiebewertung, um Ihre aktuelle Strategie zu evalieren. Experimentieren Sie, um festzustellen, wie viele Rollouts erforderlich sind, um die aktuelle Strategie hinreichend genau zu bewerten. Werten Sie dann im Rahmen der GPI (generalisierte Strategieiteration) Ihre Strategie gierig aus, um zu einer verbesserten Strategie zu gelangen. Wiederholen Sie Ihre Untersuchungen für beide Topologien des Todessterns. Lassen Sie Ihren Fighter 100 Male die ermittelte optimale Strategie fliegen: Wie oft kommt er ohne Berührungen der Wände zum Ziel? Visualisieren Sie Ihre Ergebnisse auf ansprechende Art und Weise.

PROJEKTVORSCHLAG 9:
Segeln Lernen

Schwerpunktthema: Sarsa

Personenanzahl: 2

Mindestvoraussetzung: VL-Kapitel 9

In Abbildung 7 ist ein See einer vereinfachten Form einer Gitterwelt dargestellt. Die An- und Ablegestellen für Boote sind als Startzustand S (Ufer an der linken Seite) und Zielzustand G (die zu erreichende Insel) eingetragen. Ziel ist, ausgehend von S den Zielzustand G, zu betreten. Ihnen stehen die vier standardmäßigen Aktionen zur Verfügung (links, rechts, runter, hoch). Eine Besonderheit der Umgebung ist, dass in der Mitte des Sees ein starker nördlicher Wind herrscht, von dem der Agent a priori nichts weiß. Die Stärke des Windes s ist abhängig von der x-Position im Gitter und ist am südlichen Ufer des Sees abgetragen. Der Wind bewirkt, dass das Boot zusätzlich zur normalen, aktionsgetriebenen Bewegung um s Zellen in nördliche Richtung bewegt wird. Beispiel: Angenommen, Ihr Boot befindet sich in der Zelle rechts der Insel und Sie wählen die Aktion “links”, so wird sich das Boot im Folgezustand in der Zelle nördlich der Insel befinden. Bewegungen, die das Schiffe über den Rand (das Ufer) des Sees hinausbewegen würden, werden nicht ausgeführt, d.h. der Agent verharrt in solchen Fällen in derjenigen Zelle, die dem Ufer am nächsten ist.

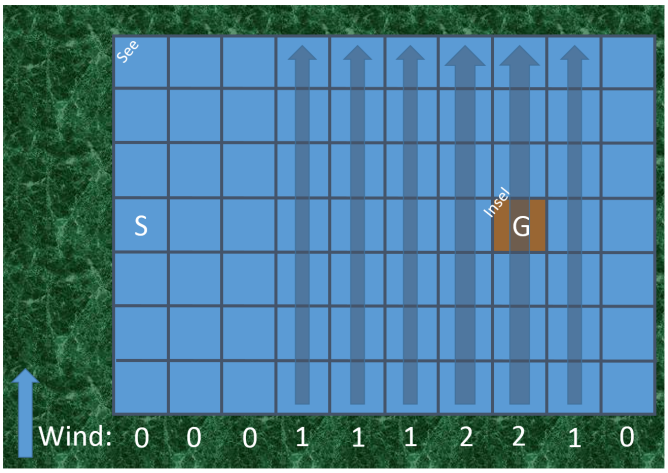

Abbildung 7: Segeln auf dem See

Die Aufgabe soll in Form einer episodischen, nicht diskontierten Modellierung als SKP-Problem realisiert werden. Kosten von 1 werden in jedem Zeitschritt vergeben, in dem das Ziel noch nicht erreicht worden ist.

(a) Verwenden Sie den aus der Vorlesung bekannten Sarsa-Algorithmus, um den Agenten in wiederholter Interaktion mit der Umgebung zu befähigen, eine gute Strategie zu erlernen. Der Agent soll dabei eine ε-gierige Strategie mit $\varepsilon = 0.1$ verwenden und als Lernrate einen Wert von $\alpha = 0.1$ benutzen. Die Q-Funktion sei zudem für alle Zustände und Aktionen zu null initialisiert ($Q(s, a) = 0 \forall s \in S, a \in A$). Wie viele Schritte umfasst die optimale Strategie für dieses Problem; wie viele Schritte benötigt die durch Ihre Implementierung gefundene Strategie?

(b) Betrachten Sie nun die folgende Erweiterung des Problems: Zusätzlich zu den genannten 4 Aktionen stehen dem Agenten vier Diagonalschritte zur Verfügung (links-hoch, rechts-hoch, links-runter, rechts-runter). Um wie viel besser ist die Strategie, die mit dieser erweiterten Aktionsmenge gelernt werden kann? Ist eine weitere Verbesserung möglich, wenn Sie ein zusätzliche neunte Aktion erlauben (Null-Aktion), bei der sich der Agent ausschließlich vom Wind treiben lässt?

(c) Erweitern Sie Ihre Lösung aus Teilaufgabe b) um stochastischen Wind. Die Stärke des Windes, sofern ungleich null, betrage nun mit einer Wahrscheinlichkeit von 1/3 genau den angegebenen Wert s, mit jeweils einer Wahrscheinlichkeit von 1/3 sei die Stärke aber $s-1$ bzw. $s+1$. Lassen Sie Ihre Implementierung mit diesen geänderten Rahmenbedingungen laufen, ermitteln Sie die optimale Strategie und visualisieren Sie sie ansprechend. Wie lange benötigt der lernfähige Agent nun, um gute Resultate zu bekommen (im Vergleich zu a) und zu b))?

PROJEKTVORSCHLAG 10:
Modellfreie Pfadsuche

Schwerpunktthema: Q-Lernen und Sarsa

Personenanzahl: 1-2

Mindestvoraussetzung: VL-Kapitel 9

Gegeben ist eine Gebäude mit 7 Räumen, dessen Grundriss in Abbildung 8 dargestellt ist. Aufgabe des Agenten ist es, durch das Gebäude zu navigieren und auf schnellstem Wege zum Zielzustand (Raum G, absorbierender Terminalzustand) zu gelangen. Alle Türen im Gebäude sind in beide Richtungen passierbar (auch wenn dies in der Abbildung nur an einer Stelle hervorgehoben ist). Dem Agenten stehen die vier gängigen Aktionen (links, rechts, hoch, runter) zur Verfügung. Zustandsübergänge (Türdurchquerungen) sollen stets mit einer Wahrscheinlichkeit von 0.9 gelingen; mit einer Wahrscheinlichkeit von 0.1 soll der Agent im aktuellen Raum verharren. Sollte der Agent eine Aktion wählen, die in einer Bewegung in eine Wand hinein resultiert, so ändert sich nichts am Zustand des Agenten. Diese Wahrscheinlichkeiten sollen dem Agenten während des Lernen aber nicht zur Verfügung stehen (lediglich die von Ihnen implementierte "Simulation" der Umgebung darf diese Werte benutzen). Für jeden Schritt erfährt der Agent Kosten von 1; ein Diskontierungsfaktor von $\gamma = 0.9$ ist zu verwenden. Sollten bei einer (gierigen) Aktionswahl zwei Aktionen gleich gut bewertet sein, so soll der Agent stets die lexikographisch kleinere Aktion wählen (also z.B. B gegenüber C vorziehen).

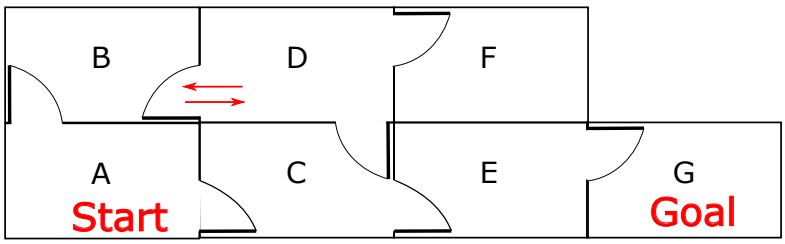

Abbildung 8: Gebäudegrundriss

(a) Implementieren Sie den Q-Lernalgorithmus und lassen Sie damit den Agenten die optimale Strategie für die gegebene Umgebung erlernen.

(b) Was ist der Erwartungswert für die Kosten der kürzesten Pfades von A nach G?

(c) Visualisieren Sie den Lernvorgang, indem Sie ein Diagramm erstellen, dass die Anzahl der Episoden den Kosten pro Episode gegenüberstellt.

(d) Testen Sie, ob Ihre Agent auch von anderen Zuständen als dem ausgewiesenen Startzustand A aus den kürzesten Weg zum Ziel findet.

(e) Variieren Sie den Wert des Diskontierungsfaktors und untersuchen Sie, ob dies etwas an der Geschwindigkeit, mit der der Agent lernt, ändert.

(f) Vergleichen Sie die von Ihnen erzielten Ergebnisse mit den Ergebnissen, die Sie unter Verwendung des Sarsa-Algorithmus erhalten.

PROJEKTVORSCHLAG 11: Einheitenbewegung

Schwerpunktthema: Q-Lernen

Personenanzahl: 2

Mindestvoraussetzung: VL-Kapitel 9

In einem Strategiespiel wollen Sie Ihre Einheiten durch einen intelligenten Agenten möglichst effizient über die Landkarte bewegen lassen. Der Agent soll in der Lage sein, für jede beliebige Kartentopologie sowie beliebige Start- und Zielpunkte, die besten Routen modellfrei zu erlernen.

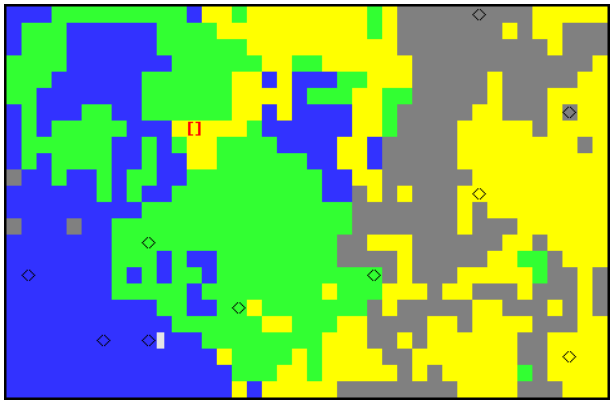

Abbildung 9: Beispiel einer zufällig generierten Landkarte

Für diese Aufgabe wird Ihnen ein kleines C-Programm zur Verfügung gestellt, das zufällige Karten in beliebiger Größe generieren, anzeigen und in Textform abspeichern kann (sh. Abbildung 9, Download im campUAS-Kurs). Das Programm gibt einen Zielpunkt (`[]`) auf der Karte vor, der angesteuert werden soll, sowie zehn Startpunkte (<>).

Es werden vier Arten von Bodenbeschaffenheiten unterschieden: Asphalt (grau), Wald (grün), Wüste (gelb) sowie Wasser (blau). Die Bewegungskosten des Agenten für einen Übergang von i nach j sind abhängig von der Bodenbeschaffenheit und sollen wie folgt angenommen werden:

$$c(i,a,j) = \frac{b(i)+b(j)}{2} \text{ wobei } \forall s \in S \text{ gilt } b(s) = \begin{cases} 1 & \text{falls Boden in } s \text{ ist Asphalt} \\ 2 & \text{falls Boden in } s \text{ ist Wald} \\ 3 & \text{falls Boden in } s \text{ ist Wüste} \\ 4 & \text{falls Boden in } s \text{ ist Wasser} \end{cases}$$

Alle Zustandsübergänge sind deterministisch; wenn der Agent an den Rand der Karte stößt, so wird kein Zustandsübergang durchgeführt. Der Agent kennt die generierte Karte a priori nicht, soll also modellfrei für eine gegebene Karte lernen, wie am effizientesten navigiert werden kann.

(a) Modellieren Sie das Problem formal. Denken Sie dabei über den Aspekt der Diskontierung nach.

(b) Ermitteln Sie die optimale Strategie für den Agenten, indem Sie modellfrei vorgehen und den Q-Lernalgorithmus verwenden. Untersuchungen Sie mögliche Vorgehensweisen bei der Exploration.

(c) Lassen Sie den Agenten für eine gegebene Kartentopologie den Q-Lernalgorithmus ausführen und nach erfolgtem Lernen von allen Startpunkten zum Zielpunkt navigieren. Ersinnen Sie eine Möglichkeit, um herauszufinden, ob der lernfähige Agent die optimale Strategie erlernt hat.

(d) Wiederholen Sie Ihre Experimente für eine größere Anzahl zufällig generierter Karten (auch in verschiedenen Größen!) und präsentieren Sie Ihre Erkenntnisee systematisch.

PROJEKTVORSCHLAG 12: Gefangen im Labyrinth

Schwerpunktthema: Wertiterationsverfahren

Personenanzahl: 2

Mindestvoraussetzung: VL-Kapitel 5

Bemerkungen zu diesem und den folgenden Projektvorschlägen: Während Sie bei allen früheren Projektvorschlägen frei in der Wahl der angewandten Software zur Lösung des Problems waren, so ist ab diesem Projektvorschlag die Nutzung der Software CLSquare[4] oder der Bibliothek SLM[5] empfohlen. Die folgenden Erläuterungen und Formulierungen sind auf CLSquare hin formuliert, lassen sich aber leicht auf SLM übertragen.

In diesem Projekt geht es um einen Agenten, der in einem 11x11 Felder großen Labyrinth gefangen ist und den Ausgang finden soll. Dieser Ausgang (quasi der Zielzustand für den Agenten) befindet sich in Zelle (8,2). Dem Agenten stehen stehen vier Aktionen zur Verfügung:

[4] CLS^2 steht für Closed Loop Simulation System und ist eine freie Software zur Durchführung von Lernexperimenten insbesondere im Kontext des dynamischen Programmierens und des Reinforcement Learning. Programmiersprache: C++

[5] PyTorch-basierende RL-Bibliothek, die zum Arbeiten mit dem Buch "Foundations of Deep Reinforcement Learning" von L. Kresser und W. Keng frei verfügbar ist. Programmiersprache: Python

- 0 (nach links gehen)
- 1 (nach rechts gehen)
- 2 (nach oben gehen)
- 3 (nach unten gehen)

Leider führt der Agent jede seiner Aktionen nur mit einer Wahrscheinlichkeit von $p = 0.8$ aus. Mit der Wahrscheinlichkeit von $1 - p$ wird er in eine zufällig ausgewählte andere Richtung zu gehen versuchen, auch wenn sich in dieser Richtung eine Wand befindet (also mit Wahrscheinlichkeit von $\frac{1}{15}$ in jede der anderen drei Richtungen). Ziel des Agenten soll es sein, den Ausgang (Zielzustand) von jeder anderen Zelle aus in minimaler Zeit zu finden. Mit "minimaler Zeit" ist hierbei gemeint, dass der Agent die durchschnittliche Anzahl von Schritten minimieren soll, die er zum Verlassen des Labyrinths benötigt.

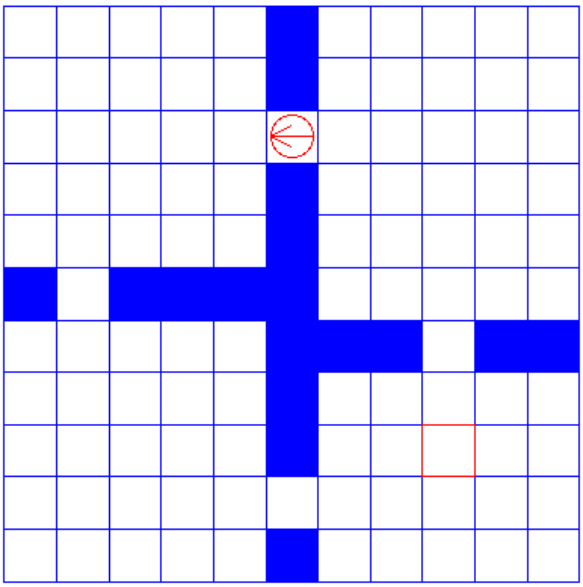

Abbildung 10: Labyrinth mit Zielzustand und beispielhafter Position des Agenten

Der Aufbau des Labyrinths ist in der Datei `maze.def` vorgegeben. Eine CLSquare-Demo finden Sie unter `clsquare/demos/Maze`.

(a) Machen Sie sich mit der Nutzung von CLSquare vertraut. Nutzen Sie Frameview für die grafische Ausgabe und zur Visualisierung Ihres Problems. Implementieren Sie testweise einen Agenten, der durch das Labyrinth navigieren und das Ziel finden kann. Hierbei ist es egal, welchen Algorithmus Sie implementieren (darf gern sehr einfach gehalten sein). Testen Sie dabei auch das Statistikmodul von CLSquare, um Ergebnisse zu protokollieren.

(b) Ein "Bewertungsexperiment" besteht aus 104 Läufen Ihres Algorithmus, bei denen Ihr Agent aus jeder der möglichen Zellen des Labyrinths gestartet wird. Ermitteln Sie die Fähigkeiten Ihres Agenten, indem Sie 10 Wiederholungen eines Bewertungsexperiments durchführen und Durchschnittswerte sowie Standardabweichungen für die Anzahl benötigter Schritte zum Ziel ermitteln. Modifizieren Sie auch testweise die Zielposition im Labyrinth, indem Sie die Datei `maze.def` editieren. Welchen Einfluss hat das auf Ihren Testagenten.

(c) Formulieren Sie nun die Lernaufgabe des Agenten im Labyrinth als stochastisches Kürzester-Pfad-Problem. Geben Sie dabei die Komponenten des zugehörigen MDP in mathematischer Notation an (S, A, p, c).

(d) Implementieren Sie den Wertiterationsalgorithmus und berechnen Sie die optimale Wertfunktion $V^\star : S_{maze} \to \mathbb{R}$ für den Labyrinth-MDP.

(e) Demonstrieren Sie die Fähigkeiten Ihres lernfähigen Agenten, indem Sie eine Lernverlaufskurve für Ihr Lernexperiment erzeugen[6].

PROJEKTVORSCHLAG 13: Aufschwingen lernen: Pole

Schwerpunktthema: Q-Lernen

Personenanzahl: 2

Mindestvoraussetzung: VL-Kapitel 9

Bemerkung: Bitte beachten Sie zunächst die allgemeinen Bemerkungen, die zu Beginn des vorangegangenen Projektvorschlags gegeben sind.

Beim Pole handelt es sich um ein Problem, bei dem ein frei schwingbares Pendel aufgeschwungen werden soll. Der Arbeitsbereich (Zustandsraum) des Pole-Systems ist zweidimensional und umfasst:

- die Winkelstellung des Pendels (von -180 bis 180 Grad, die 12-Uhr-Stellung gilt als Nullstellung)
- die Winkelgeschwindigkeit des Pendels

Dem Agenten stehen nur zwei Aktionen zur Verfügung, mit denen er das Pendel nach links (4N) bzw. nach rechts (ebenfalls um 4N) beschleunigen kann. Als erfolgreich abgeschlossen (erfolgreiches Aufschwingen) gilt eine Episode, wenn der Stab in eine Position

[6] Eine Lernverlaufskurve protokolliert die Leistungsfähigkeit der Strategie des Agenten während des Lernvorgangs. An der x-Achse lassen sich daher beispielsweise die Anzahl der Aktualisierungen an der Wertfunktion abtragen, an der y-Achse entsprechend die Anzahl der durchschnittlich benötigten Schritte, um zum Zielzustand zu gelangen. Um solche eine Grafik zu erzeugen, muss der Lernvorgang also regelmäßig zwischendrin pausiert und durch Testläufe zur Qualitätserhebung unterbrochen werden. Verwenden Sie das Werkzeug gnuplot, um Ihre Lernverlaufskurve zu zeichnen.

gebracht worden ist, die um weniger als 0.1rad (also ca. 6 Grad) von der Nullstellung abweicht. Die Dynamik der Umgebung ist bereits in CLSquare implementiert und kann einfach verwendet werden. Für SLM ist eine Adaption der Car-Pole-Umgebung erforderlich, die den Wagen stillstehen lässt.

Aufgabenstellung

(a) Machen Sie sich mit der Nutzung von CLSquare/SLM vertraut. Nutzen Sie Frameview für die grafische Ausgabe und zur Visualisierung Ihres Problems. Implementieren Sie testweise einen Agenten, der nach links und rechts beschleunigen und ggf. sogar das Pendel aufschwingen kann. Hierbei ist es egal, welchen Algorithmus Sie implementieren (darf gern sehr einfach gehalten sein). Testen Sie dabei auch das Statistikmodul von CLSquare, um Ergebnisse zu protokollieren.

(b) Ein "Bewertungsexperiment" besteht aus 1000 Läufen Ihres Algorithmus, bei denen Ihr Agent jeweils von einer der Startpositionen aus der Startpositionsmenge ($S_{startset} = \{(\alpha, \omega) | \alpha = \frac{k\pi}{20}, k = -19, \ldots, 20, \omega \in \{-0.5, 0, 0.5\}\}$) aus gestartet wird und möglichst schnell das Pendel aufschwingen soll. Ermitteln Sie die Fähigkeiten Ihres Agenten, indem Sie über die Wiederholungen des Bewertungsexperiments Durchschnittswerte sowie Standardabweichungen für die Anzahl der Schritte bis zum Aufschwingen ermitteln.

(c) Implementieren den Q-Lernalgorithmus für den von Ihnen diskretisierten Zustandsraum des Pole-Problems.

(d) Demonstrieren Sie die Fähigkeiten Ihres lernfähigen Agenten, indem Sie eine Lernverlaufskurve für Ihr Lernexperiment erzeugen[7].

PROJEKTVORSCHLAG 14: Balancieren lernen: Cart Pole

Schwerpunktthema: Q-Lernen

Personenanzahl: 2

Mindestvoraussetzung: VL-Kapitel 9

Bemerkung: Bitte beachten Sie zunächst die allgemeinen Bemerkungen, die zu Beginn des vorangegangenen Projektvorschlags gegeben sind.

[7] Eine Lernverlaufskurve protokolliert die Leistungsfähigkeit der Strategie des Agenten während des Lernvorgangs. An der x-Achse lassen sich daher beispielsweise die Anzahl der durchgeführten Episoden oder aber die Anzahl der Aktualisierungen der Q-Funktion abtragen, an der y-Achse entsprechend die Anzahl der durchschnittlich benötigten Schritte, um in die Zielregion zu gelangen. Um solche eine Grafik zu erzeugen, muss der Lernvorgang also regelmäßig zwischendrin pausiert und durch Testläufe zur Qualitätserhebung unterbrochen werden. Verwenden Sie das Werkzeug gnuplot, um Ihre Lernverlaufskurve zu zeichnen.

Aufgabenstellung Beim Cart Pole handelt es sich um ein Problem, bei dem ein frei schwingbarer Stab aufgeschwungen und balanciert werden soll. Im Gegensatz zu Projektvorschlag VI ist der Stab jedoch nicht fest montiert, sondern an einem kleinen Wagen angebracht, der nach links und rechts beschleunigt werden kann. In diesem Projekt betrachten wir nur die Aufgabe des Balancierens; die gegebenfalls vorweg ausführbare Aufgabe des Aufschwingens lassen wir außer Acht.

Der Arbeitsbereich (Zustandsraum) des Cart-Pole-Systems ist vierdimensional und umfasst:

- die Winkelstellung des Pendels (von -0.7 rad bis 0.7 rad, die 12-Uhr-Stellung gilt als Nullstellung, bei Werten darüber bzw. darunter gilt das Pendel als umgefallen)
- die Winkelgeschwindigkeit des Pendels
- die Position des Wagens (von -2.4m bis 2.4m)
- die Geschwindigkeit des Wagens

Als Zielbereich, in dem das System gehalten werden soll (in dem es entsprechend Belohungen gibt) ist so definiert, dass die Winkelstellung zwischen -0.05 und 0.05rad liegen und der Wagen nicht mehr als 5cm nach links oder rechts von seiner Mittelstellung abweichen soll. Die Startsituationen des Agenten sollen so definiert sein, dass sie zufällig aus dem Bereich ausgewählt werden, in dem die Winkelstellung vom Betrag her kleiner als 0.5rad ist und gleichzeitig die Position des Wagens nicht mehr als 50cm von der Mittelstellung abweicht (beide Geschwindigkeiten sind 0).

Dem Agenten stehen nur zwei Aktionen ($A = \{-10, 10\}$) zur Verfügung:

- -10 (Wagen mit 10N nach links beschleunigen)
- 10 (Wagen mit 10N nach rechts beschleunigen)

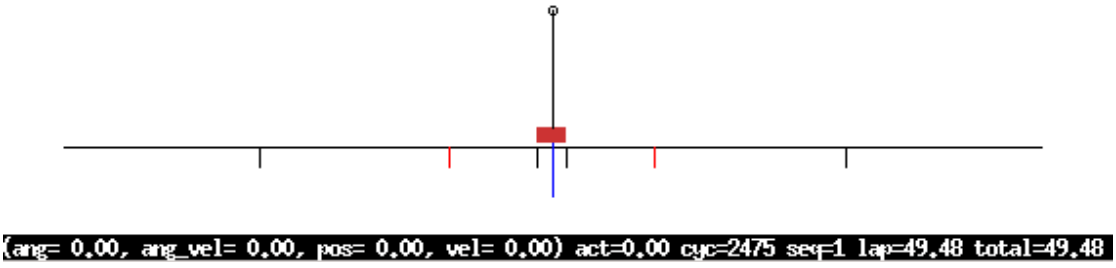

Abbildung 11: Screenshot vom Cart-Pole-System mit beispielhaftem Zustand des Agenten

(a) Machen Sie sich mit der Nutzung von CLSquare/SLM vertraut. Nutzen Sie Frameview bzw. das OpenAI Gym für die grafische Ausgabe und zur Visualisierung Ihres Problems. Implementieren Sie testweise einen Agenten, der nach links und rechts beschleunigen und ggf. sogar für einige Zeit erfolgreich balancieren kann. Hierbei ist es egal, welchen Algorithmus Sie implementieren (darf gern sehr einfach gehalten sein). Testen Sie dabei auch das Statistikmodul von CLSquare, um Ergebnisse zu protokollieren.

(b) Ein "Bewertungsexperiment" besteht aus 1000 Läufen Ihres Algorithmus, bei denen Ihr Agent jeweils von einer der Startpositionen aus gestartet wird und sich für 200 Zeitschritte beweisen muss (wenn er in dieser Zeit den Stab nicht hat fallen lassen, gilt die Episode als Erfolg). Ermitteln Sie die Fähigkeiten Ihres Agenten, indem Sie über die Wiederholungen des Bewertungsexperiments Durchschnittswerte sowie Standardabweichungen für die Anzahl der Schritte, in denen der Agent erfolgreich balancierte, ermitteln.

(c) Implementieren den Q-Lernalgorithmus für den von Ihnen diskretisierten Zustandsraum des Cart-Pole-Problems.

(d) Experimentieren Sie mit dem Einsatz eines fortgeschritteneren Funktionsapproximator (z.B. künstlicher neuronaler Netze). In CLSquare ist dafür die Bibliothek n++ enthalten, bei SLM-Nutzung können Sie die PyTorch-API Net verwenden.

(e) Demonstrieren Sie die Fähigkeiten Ihres lernfähigen Agenten, indem Sie eine Lernverlaufskurve für Ihr Lernexperiment erzeugen[8].

[8] Eine Lernverlaufskurve protokolliert die Leistungsfähigkeit der Strategie des Agenten während des Lernvorgangs. An der x-Achse lassen sich daher beispielsweise die Anzahl der durchgeführten Episoden oder aber die Anzahl der Aktualisierungen der Q-Funktion abtragen, an der y-Achse entsprechend die Anzahl der durchschnittlich benötigten Schritte, um in die Zielregion zu gelangen. Um solche eine Grafik zu erzeugen, muss der Lernvorgang also regelmäßig zwischendrin pausiert und durch Testläufe zur Qualitätserhebung unterbrochen werden. Verwenden Sie das Werkzeug gnuplot oder die SLM-spezifischen Werzeuge, um Ihre Lernverlaufskurve(n) zu zeichnen.