Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Institut für Anthropomatik (IFA)
Lehrstuhl Prof. Dr.-Ing. Rüdiger Dillmann

Robotik II SS 2009

2. Übungsblatt

Prof. Dr.-Ing. R. Dillmann
Haid-und-Neu-Str. 7, Geb. 07.21
D-76131 Karlsruhe

Dipl. Inform. Sven R. Schmidt-Rohr
Dipl. Inform. Rainer Jäkel
http://wwwiaim.ira.uka.de

Aufgabe 1 (Planungsverfahren)

Ein Serviceroboter befindet sich in einer Küche und dort befinden sich auch eine **Kiste**, ein **Kühlschrank** und eine **Flasche** Bier. Desweiteren hat der Serviceroboter eine **Roboterhand**, es gibt einen freien Platz **Neben_Kühlschrank** und ein offener Kühlschrank kann als **Kühlschrank_Auf** modelliert werden. Hierbei ist Kühlschrank_Auf als Objekt anzusehen. Schließlich hat der Serviceroboter einen Besitzer, der eine **Besitzerhand** hat. Objekte könnten für den Roboter **Frei** zum Zugriff sein, sich **Inhand** befinden oder ein Objekt **Blockiert** ein anderes. Der Roboter kann nun ein freies Objekt **Greife**, oder von einem blockierten Objekt **Greife_Von**. Schließlich kann der Roboter ein Objekt in seiner Hand an einen anderen Ort **Bewege**.

a) Modellieren Sie die Aktionen Domäne dieses Szenarios in STRIPS-Notation. Beachten Sie, dass für STRIPS gilt: Nur positive Literale werden explizit angegeben, alle negativen Literale werden nicht explizit angegeben (außer im Effekt) und nicht angegebene Literale werden automatisch als negativ angenommen (Annahme der abgeschlossenen Welt). Ziele sind Konjunktionen von funktionsfreien Atomen. Aktionseffekte sind Konjunktionen von Prädikaten.

b) Nun befindet sich die Kiste vor dem Kühlschrank, blockiert also den direkten Zugang zu diesem, die Flasche steht im Kühlschrank, wird also von diesem blockiert. Alle weiteren Objekte sind frei. Modellieren Sie den Zustand (Start) in STRIPS-Notation.

c) Der Serviceroboter soll nun den Wunsch seines Besitzers "Hol mir mal ne Flasche Bier" erfüllen. Modellieren Sie ein passendes Ziel in STRIPS-Notation.

d) Erstellen Sie nun manuell einen kurzen Plan vom Start aus, welcher das Ziel erreicht. Notieren Sie alle Aktionen und Zwischenzustände.

e) Welche Probleme können auftreten, wenn der Roboter eine mechanische, stapelbasierte Vorwärtssuche durchführen würde?

f) Welche Probleme im Szenariokontext können durch die Domänensemantik enstehen?

Aufgabe 2 (Planungsverfahren II)

Ein Serviceroboter soll einen Tisch decken mit folgenden Objekten: **Teller**, **Tasse**, **Untertasse**, **Löffel**, **Serviette** und **Matte**. Es gelten folgende Regeln: die Tasse kommt auf die

Untertasse, der Löffel auf die Serviette, die Matte unter alles andere. Der Roboter hat nun zwei Hände, mit denen er Objekte auf den Tisch stellen kann, daher bietet sich POP Planung an.

a) Modellieren Sie einen allgemeinen POP-Vorranggraph für das Tischdecken.

b) Linearisieren Sie den Graphen zu einem schnellen Plan unter Berücksichtigung der zwei Roboterhände, mit denen maximal zwei parallelisierbare Aktionen zur Laufzeit auch tatsächlich parallel ausgeführt werden können.

c) Ein anderer Serviceroboter hat nun sehr klobige Hände und kann die Serviette nicht mehr aufbringen, nachdem der Teller schon auf dem Tisch steht. Lösen Sie den Konflikt auf, indem Sie den Graph anpassen.

d) Um ein größeres Szenario zu modellieren, soll nun auf HTNs gewechselt werden. Modellieren Sie den Plan aus a) in einem HTN-Baum.

e) Der Roboter soll nun erst seinem Besitzer eine Flasche Bier holen, dann den Tisch decken und schließlich noch das Zimmer sauber machen. Machen Sie sich klar, wie leicht die Erweiterung mittels HTNs funktioniert.

Aufgabe 3 (Markov Decision Processes)

a) Implementieren Sie die diskrete MDP Value Iteration in einer Programmiersprache ihrer Wahl. Ein CLI Programm ist ausreichend. Im Folgenden der Pseudocode mit der *value function* V, dem Belohnungsmodell R und dem Transitionsmodell T:

MDPdiscreteValueIteration():
    $V(s) \leftarrow min(R)$
    repeat until max step number
        for each state s **in** S **do**
            $V(s) \leftarrow \gamma max_u(R(s,u) + \sum_{s'} T(s',u,s)V(s'))$
        endfor
    end repeat
    return V

MDPbestAction(s):
    return $argmax_u(R(s,u) + \sum_{s'} T(s',u,s)V(s'))$

Beachten Sie, dass V ein Vektor, R eine Matrix und T ein Tensor 3. Stufe ist. R(s, u) bezeichnet den Matrixeintrag in Zeile s und Spalte u und ist demnach ein Skalar. T(s', u, s) bezeichnet den Tensoreintrag bei s', u und s und ist demnach auch ein Skalar. Die Bezeichnung max_u bedeutet, dass nur der Wert des Ausdrucks in der Klammer für ein zugehöriges u genommen wird, bei dem der Ausdruck den maximalen Wert annimmt. Es wird also in jedem Iterationsschritt in 3 verschachtelten Schleifen über s, u (für max) und s' (für die innere Summe) iteriert. Für die folgenden Aufgaben ist das Eintragen der Modelldaten direkt in den Quellcode ausreichend, es wird also keine spezielle Einlesemethode benötigt.

b) Berechnen Sie nun die utilities der Zustände der value function $V(s)$ für das Beispiel aus der Vorlesung für $\gamma = 0.8$, $\gamma = 0.9$ und $\gamma = 0.95$ über 40 Schritte. Das Beispiel aus der Vorlesung hat die folgenden Modelldaten:

$T_a(S', U, S)$:

U_1:

	S'_1	S'_2	S'_3
S_1 :	0.75	0.25	0.0
S_2 :	0.0	0.25	0.75
S_3 :	0.25	0.25	0.5

U_2:

	S'_1	S'_2	S'_3
S_1 :	0.75	0.25	0.0
S_2 :	0.25	0.75	0.0
S_3 :	0.50	0.25	0.25

U_3:

	S'_1	S'_2	S'_3
S_1 :	0.75	0.25	0.0
S_2 :	0.25	0.75	0.0
S_3 :	0.25	0.50	0.25

$R_a(S, U)$:

	U_1	U_2	U_3
S_1 :	-1	-1	0
S_2 :	-1	-1	0
S_3 :	-1	4	0

Die Semantik war:
S_1: kein Mensch anwesend
S_2: Mensch anwesend, ohne Intention zu folgen
S_3: Mensch anwesend, mit Intention zu folgen
U_1: Roboter sagt *folge mir*
U_2: Roboter führt an einen Ort
U_3: Roboter macht nichts

Semantik des Belohnungsmodells: Sprechen und drehen kosten -1, Nichtstun kostet nichts, der Versuch, einen Menschen zu führen gibt eine Belohnung von 5.

c) Vergleichen Sie die Ergebnisse und machen Sie sich die je nach γ unterschiedlich schnelle Kontraktion klar. Wieso ist die utility $V()$ von s_3 immer am größten?

d) Wieso ist der relative Unterschied zwischen $V(s_1)$ und $V(s_3)$ bei $\gamma = 0.8$ größer, als bei $\gamma = 0.9$ und jener größer, als bei $\gamma = 0.95$?

e) Wie sieht es mit der absoluten Differenz aus (Lassen Sie dafür die value function über 100 Schritte berechnen, um dem Grenzwert möglichst nahe zu kommen) ?

f) Berechnen Sie nun die value function für leicht geänderte Transitionsmodelle (das Belohnungsmodell bleibt R_a, nur die fett gedruckten Zahlen haben sich geändert) für $\gamma = 0.8$:

$T_b(S', U, S)$:

U_1:

	S'_1	S'_2	S'_3
S_1 :	0.75	0.25	0.0
S_2 :	0.0	**0.74**	**0.26**
S_3 :	0.25	0.25	0.5

U_2:

	S'_1	S'_2	S'_3
S_1 :	0.75	0.25	0.0
S_2 :	0.25	0.75	0.0
S_3 :	0.50	0.25	0.25

U_3:

	S'_1	S'_2	S'_3
S_1 :	0.75	0.25	0.0
S_2 :	0.25	0.75	0.0
S_3 :	0.25	0.50	0.25

$T_c(S', U, S)$:

U_1:

	S'_1	S'_2	S'_3
S_1 :	0.75	0.25	0.0
S_2 :	0.0	**0.76**	**0.24**
S_3 :	0.25	0.25	0.5

U_2:

	S'_1	S'_2	S'_3
S_1 :	0.75	0.25	0.0
S_2 :	0.25	0.75	0.0
S_3 :	0.50	0.25	0.25

U_3:

	S'_1	S'_2	S'_3
S_1 :	0.75	0.25	0.0
S_2 :	0.25	0.75	0.0
S_3 :	0.25	0.50	0.25

Im Vergleich zu T_a, wird ein Mensch bei T_b nun deutlich seltener dem Roboter folgen wollen, wenn er dazu aufgefordert wurde.

g) Vergleichen Sie die utilities der Zustände $V(s)$ von T_a und T_b. Beschreiben Sie in Worten, was hier geschieht, wenn man die Modelländerung im semantischen Kontext betrachtet.

h) Vergleichen Sie nun nicht nur die utilities der value function von $V_{(T_b)}(S_2)$ und $V_{(T_c)}(S_2)$, sondern auch die zugehörige, optimale Handlung (also die policy). Welcher einfache, aus dem Belohnungsmodell R_a ableitbare, numerische Zusammenhang führt zu diesem Effekt gerade zwischen diesen beiden Belohnungsmodellen T_b und T_c?

i) Wie kann das Belohnungsmodell R_a, mit einer minimalen ganzzahligen Änderung (beachte Belohnungssemantik), in ein Belohnungsmodell R_b abgewandelt werden, so dass die optimalen Handlungen der policy für T_b und T_c identisch sind?

Aufgabe 4 (Partially Observable Markov Decision Processes)

Der MDP aus der vorigen Aufgabe soll nun zu einem POMDP erweitert werden. Zur Vereinfachung sollen die Beobachtungen M direkt an Zustände gekoppelt werden, mit der Semantik Beobachtung $m_i :=$ RoboterGlaubtZustandZuBeobachten(s_i). Die Beobachtungen bilden die Zeilen und die Zustände die Spalten der Matrix des Beobachtungsmodells. Dabei ist zu beachten, dass sich Spalten auf 1 summieren müssen (da die Wahrscheinlichkeiten, wie sich ein realer Zustand auf Beobachtungen verteilt, 1 ergeben muss). Die Zeilen müssen sich aber nicht auf 1 aufsummieren, da es durchaus eine allgemeines, stärkeres Vorkommen von bestimmten Beobachtungen geben kann, die andere Beobachtungen im Vergleich zur Realität zu stark dominieren (Bias).

Die Beobachtungsfähigkeiten des Roboters haben nun folgende Eigenschaften:
In 30% der Fälle, wenn kein Mensch anwesend ist, meint der Roboter in Bildgebungsartefakten einen zu erkennen, er glaubt dann jedoch nie, dass dieser schon die Intention hätte zu folgen, da kein Dialog stattfand.
Ist ein Mensch anwesend, übersieht ihn der Roboter nur, wenn noch kein Dialog stattfand und noch keine Intention zu folgen vorhanden ist und dann auch nur in 10% der Fälle.
Aufgrund eines fehlenden Gedankenlesesensors und Problemen der Dialogführung, ist die Einordnung der Intention eines vorhandenen Menschen, ob er folgen möchte, oder nicht, besonders fehleranfällig. Hat der Mensch nicht die Intention zu folgen, trifft der Roboter, unter der Annahme, dass er die Anwesenheit korrekt erkannt hat, nur in 2 von 3 Fällen die richtige Annahme bezüglich der Intention. Hat der Mensch dagegen die Intention zu folgen, ist auch der Roboter in 80% der Fälle davon überzeugt.

a) Erstellen sie die Matrix des Beobachtungsmodells O_a aus den angegebenen Fähigkeiten. Welcher allgemeine, sensorische Bias ist sichtbar (Summe der jeweiligen Zeile bilden)?

Da approximative POMDP value iteration Algorithmen wesentlich komplexer sind als MDP value iteration, soll an dieser Stelle auf eine Implementierung verzichtet werden (siehe jedoch Zusatzaufgabe). Mit PBVI wurde nun eine policy für das vorliegende Modell (T_a, R_a, O_a)

berechnet, mit $\gamma = 0.98$ bis zur Horizonttiefe 8 und mit einer Sample-Anzahl von ~ 30[1]. Die policy Γ besteht aus folgenden linearen Funktionen α im Format $u : (V(s_1), V(s_2), V(s_3))$:

$\alpha_1 = u_2$: (-0.53,0.63,4.69)
$\alpha_2 = u_3$: (0.61,1.27,1.10)
$\alpha_3 = u_1$: (-0.92,2.66,1.46)
$\alpha_4 = u_2$: (-0.68,0.19,5.18)
$\alpha_5 = u_2$: (-0.72,0.07,5.24)
$\alpha_6 = u_3$: (0.45,1.62,1.30)
$\alpha_7 = u_1$: (-1.78,2.69,1.20)
$\alpha_8 = u_1$: (-0.99,2.66,1.45)

b) Die Reihenfolge ist jene, in der sie vom Algorithmus produziert wurde. Was fällt bei der Gruppierung in Funktionen mit gleicher zugehöriger Handlung auf?

c) Implementieren Sie nun die Methode, welche die optimale Handlung für eine beliebige Zustandsvermutung aus der policy Γ extrahiert:

POMDPbestAction(b, Γ):
$\alpha_{max} \leftarrow argmax_{\alpha}((\alpha \in \Gamma) \cdot b)$
return $u(\alpha_{max})$

Beachten Sie: Γ ist eine Liste von Paaren (u, α), wobei u die zu α gehörige Handlung ist. α ist eine lineare Funktion mit Dimension $|S|$. Die Zustandsvermutung (belief) b hat ebenfalls die Dimension $|S|$. Die Methode gibt einfach die Handlung vom zugehörigen α mit dem höchsten Wert in b (Skalarprodukt) zurück.

d) Berechnen Sie nun die optimale Handlung für
$b_1 = (1, 0, 0)$
$b_2 = (0, 1, 0)$
$b_3 = (0, 0, 1)$
Vergleichen Sie diese Ergebnisse mit den Ergebnissen des MDP aus der vorigen Aufgabe. Was fällt auf?

e) Berechnen Sie die optimale Handlung für
$b_4 = (0.33, 0.34, 0.33)$.
Die Zustandsvermutung b_4 spiegelt eine (bis auf die kleine Abweichung durch die Rundung) völlige Gleichverteilung und damit maximales Unwissen über den momentanen, realen Zustand der Welt wider. Wie erklären Sie sich das Ergebnis der Entscheidung aus der Semantik von Beobachtungs-, Belohnungs-, und Transitionsmodell?

f) Erscheint Ihnen diese Entscheidung in b_4 in einem realen Kontext sinnvoll und wenn nicht, wie könnte das Belohnungsmodell geändert werden, damit ein intuitiveres Verhalten entsteht?

Zusatzaufgabe 1 (Point Based Value Iteration)

[1] Falls Sie PBVI nachprogrammieren, beachten Sie, dass keine völlig deterministischen Ergebnisse entstehen und Entscheidungsfunktionen mit identischen Startparametern durchaus voneinander abweichen können. Bei höherer Genauigkeit sind diese Abweichungen bei der Berechnung optimaler Handlungen jedoch nicht mehr relevant.

Die Zusatzaufgabe soll dem Verständnis jenseits des Pflichtstoffs helfen. Sie hat einen deutlich höheren Schwierigkeitsgrad, bitte konsultieren Sie daher zur Lösung zusätzlich auch die in der Vorlesung angegebene Orginalliteratur. Implementieren Sie den Point Based Value Iteration Algorithmus in einer Programmiersprache ihrer Wahl. Der Pseudocode lautet:

getPBVIpolicy(Expandsteps, Horizonstepsize)

$O = Observationmodel(Measurements, States)$
$T = Transactionmodel(Newstates, Actions, Oldstates)$
$R = Rewardmodel(States, Actions)$
$B = initialDistributedSampleSet()$
$\Gamma = ()$
for n = 1 to Expandsteps do
  for t = 1 to Horizonstepsize do
    $\Gamma' = ()$
    for each α^k in Γ do
      for each a in Actions do
        for each m in Measurements do
          for $dim = 1$ to $|States|$ do
            $v^k_{a,m,dim} = \sum_i^{|States|} \alpha^k_i * Prob(O[m,i]) * Prob(T[i,a,dim])$
          endfor
        endfor
      endfor
    endfor
    for each b in B do
      for each a in Actions do
        for each m in Measurements do
          for each α'''^k in Γ do
            $val_{\alpha'''^k_m} = \sum_{dim}^{|States|} v^k_{a,m,dim} * b_{dim}$
          endfor
          $kmax_{b,a} = max(val_{*,m})$
          for $dim = 1$ to $|States|$ do
            $\alpha''_{dim} += v^{kmax}_{a,m,dim}$
          endfor
        endfor
        for $dim = 1$ to $|States|$ do
          $\alpha'_{b,a,dim} = \gamma * (R(dim,a) + \alpha''_{dim})$
          $val_{\alpha'_{b,a},b} += \gamma * (R(dim,a) + \alpha''_{dim}) * b_{dim}$
        endfor
      endfor
      $\alpha_b = argmax(val_{\alpha'_{b,*},b})$
      if $\alpha_b \notin \Gamma'$ then
        $\Gamma' \leftarrow \alpha_b$
      endif
    endfor
    $\Gamma = \Gamma'$

40: endfor
41: for each b in B do
42: for each a in Actions do
43: $s = random - multinomial(b)$
44: $s' = random - multinomial(T(s, a, *))$
45: $m = random - multinomial(O(s', *))$
46: for $dim = 1$ to $|States|$ do
47: $B'_{a,dim} = \frac{\sum_i^{|States|} O[m,i]*T[i,a,dim]*b_i}{normalizer}$
48: endfor
49: endfor
50: for $b_i = 1$ to $|B'[*]|$ do
51: for $b'_j = 1$ to $|B'[*]|$ do
52: $norm_{i,j} = |b_i - b'j|$
53: endfor
54: $minnorm_i = min(norm_{i,*})$
55: endfor
56: $maxb = max(minnorm_*)$
57: $B \leftarrow b'_{maxb}$
58: endfor
59: endfor
60: return Γ
end

Man beachte die beiden unabhängigen Schritte *backup* von Zeile 7 bis 40 und *sample set expansion* von Zeile 41 bis 58. Der *backup* Schritt erzeugt erst temporäre Koeffizienten für lineare Funktionen in Zeile 9-17. Anschließend wird auf deren Basis in jedem *belief sample point* die Funktion mit dem besten Gewinnerwartungswert berechnet (Zeilen 18-38). Der *sample set expansion* Schritt würfelt zuerst neue *belief points* von jedem aktuellen b ausgehend auf Basis der stochastischen Modelle aus (Zeilen 42-49) und bestimmt anschließend von all jenen neuen Punkten den, der von allen bisherigen Punkten am weitesten entfernt ist (Zeilen 50-56). Durch dieses Verfahren erreicht die *sample set expansion* nur diejenigen Bereiche des Zustandsvermutungsraum, die im Szenario tatsächlich erreicht werden können und gleichzeitig sind die Punkte sehr stark gleichverteilt. Die Schritte *backup* (erhöht den Horizont) und *sample set expansion* (erhöht die Sampledichte) können beliebig gemischt werden, die Genauigkeit der policy kann nur steigen, da auch dieser Algorithmus eine Kontraktion darstellt. Es gilt immer $|\Gamma| \leq |B|$.

Die korrekte Implementation kann recht komfortabel durch den Vergleich mit den Ergebnissen einer MDP value iteration verifiziert werden. Hierzu sind die utilities der POMDP policy in den Eckpunkten des Zustandsvermutungssimplex $(0, ..., 0, 1, 0, ..., 0)$ mit den utilities der MDP policy zu vergleichen.