Markov Decision Processes (MDPs) and Reinforcement Learning (RL)

Sven Koenig, USC

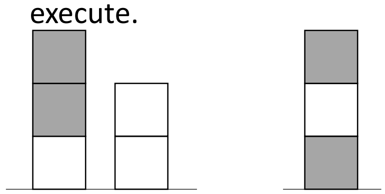
Russell and Norvig, 3rd Edition, Sections 17.1-17.2

These slides are new and can contain mistakes and typos.
Please report them to Sven (skoenig@usc.edu).

1

Decision-Theoretic (= Probabilistic) Planning

- Blocks World with 3 changes
  - Blocks are either white or black, rather than named.
  - The standard move actions can go wrong with probability 0.4, in which case the moved block slips during the move and ends up on the table. If the move actions work as intended, they take 2 minutes to execute. If they go wrong, they take one minute to execute.
  - There are also paint actions that paint any given block either white or black without moving them. They always work as intended and take 3 minutes to execute.

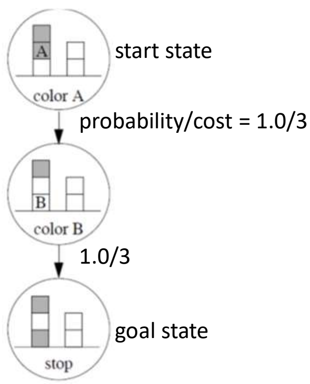


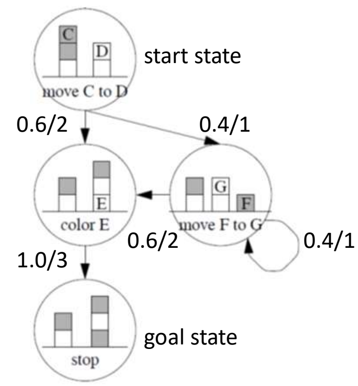
Start state Goal state

Note
- the current state is always known
- action executions can result in several outcomes
- a probability distribution over these outcomes is known
- this is a generalization of deterministic search
- we continue to assume that action costs are always strictly positive

2

Evaluating Decision-Theoretic Plans





expected plan-execution cost (here: time)
6 minutes

expected plan-execution cost
c_1 = 5.67 minutes

3

Evaluating Decision-Theoretic Plans

This is similar to deterministic search, where we assume that the action cost and the successor state depend only on the current state and the action executed in it.

- We assume that the expected action cost and the probability distribution over the successor states depend only on the current state and the action executed in it ("Markov property"). In other words, it does not matter how the current state was reached.

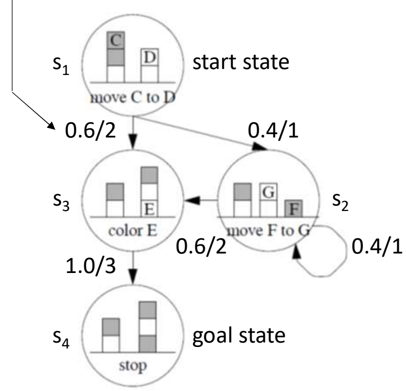
- An example that illustrates the resulting independence assumptions:
  - $p(s_{t+2}=s'' \mid a_{t+1}=a', a_t=a, s_t=s)$
    $= \Sigma_{s'}\, p(s_{t+2}=s'', s_{t+1}=s' \mid a_{t+1}=a', a_t=a, s_t=s)$
    $= \Sigma_{s'}\, p(s_{t+2}=s'' \mid a_{t+1}=a', s_{t+1}=s', a_t=a, s_t=s)\; p(s_{t+1}=s' \mid a_{t+1}=a', a_t=a, s_t=s)$
    $= \Sigma_{s'}\, p(s_{t+2}=s'' \mid a_{t+1}=a', s_{t+1}=s')\; p(s_{t+1}=s' \mid a_t=a, s_t=s)$

4

Evaluating Decision-Theoretic Plans

$p(s_{t+1}=s_3 | s_t=s_1, a_t=\text{move C to D})=0.6$ / $c(s_1,\text{move C to D},s_3)=2$

- c_i = expected plan-execution cost until a goal state is reached if one starts in state s_i and follows the plan
- $c_1 = 0.4\ (1+c_2) + 0.6\ (2+c_3)$
  $c_2 = 0.4\ (1+c_2) + 0.6\ (2+c_3)$
  $c_3 = 1.0\ (3+c_4)$
  $c_4 = 0$
- $c_1 = c_2 = 5.67$
  $c_3 = 3$
  $c_4 = 0$



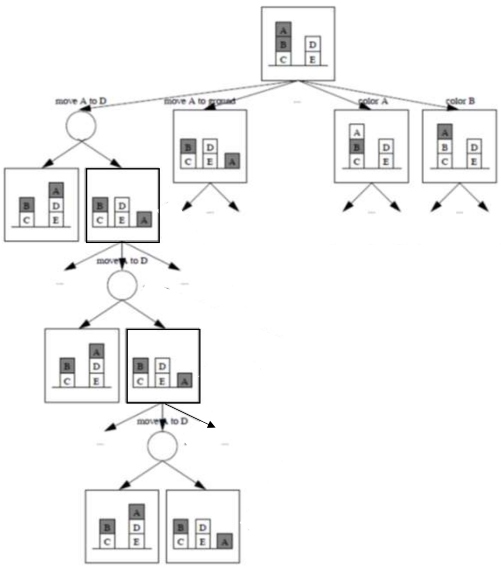
5

Evaluating Decision-Theoretic Plans

- In general, one solves the following system of equations calculating the expected plan-execution cost of decision-theoretic plans:
  - $c_i = 0$ if s_i is a goal state
  - $c_i = \Sigma_k\ p(s_k | s_i, a(s_i))\ (c(s_i, a(s_i), s_k) + c_k)$ if s_i is not a goal state
- One solves the system of equations either with Gaussian elimination (as on the previous slide) or as follows:
  - for all i
    - $c_{i,0} = 0$
  - for t=0 to ∞ ← The typical termination criterion is: $|c_{i,t+1} - c_{i,t}| < \varepsilon$ for all i (for a given small positive ε).
    - for all i
      - $c_{i,t+1} = 0$ if s_i is a goal state
      - $c_{i,t+1} = \Sigma_k\ p(s_k | s_i, a(s_i))\ (c(s_i, a(s_i), s_k) + c_{k,t})$ if s_i is not a goal state

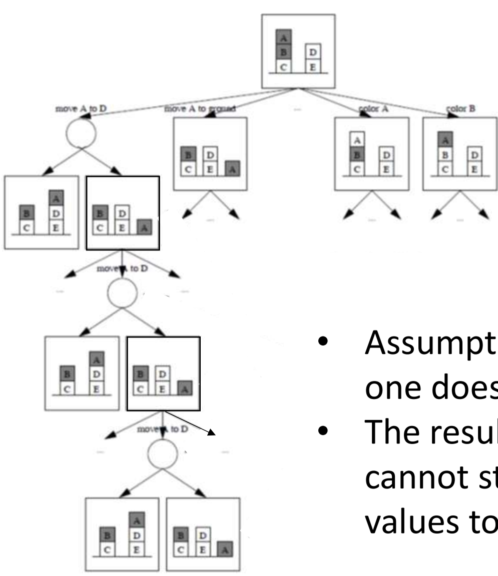
6

Decision-Theoretic Planning



7

Decision-Theoretic Planning

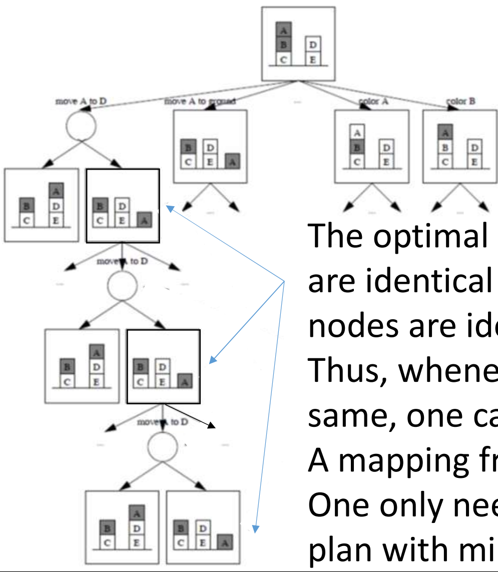


This is similar to deterministic search.

- Assumption: One has to execute actions forever if one does not reach a goal state earlier.
- The resulting decision tree is infinite. Thus, one cannot start at the utility nodes and propagate the values toward the root of the decision tree.

8

Decision-Theoretic Planning



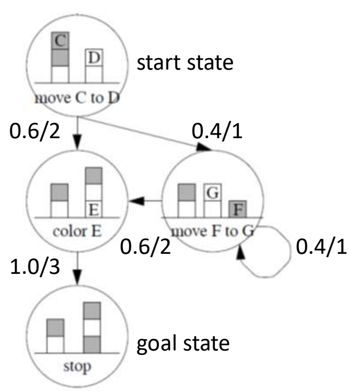
This is similar to deterministic search, where one only needs to consider all paths without cycles from the start state to a goal state to determine a plan with minimal plan-execution cost.

The optimal actions associated with these choice nodes are identical since the subtrees rooted in the choice nodes are identical.
Thus, whenever the state (= configuration of blocks) is the same, one can execute the same action.
A mapping from states to actions is called "policy."
One only needs to consider all policies to determine a plan with minimal expected plan-execution cost.

9

Decision-Theoretic Planning

- This is, for example, a policy although policies are typically written as functions that map each state to the action that should be executed in it.



10

Decision-Theoretic Planning

- In the deterministic case:
  - Out of all possible plans, we need to consider only cycle-free paths because there is always a cycle-free path that is cost-minimal. This insight dramatically reduces the number of plans that we need to consider. However, it still takes too long to consider all cycle-free paths and determine one of minimal cost. Thus, we needed to study more sophisticated search algorithms.
- In the probabilistic case:
  - Out of all possible plans, we need to consider only policies because there is always a policy that is cost-minimal. This insight dramatically reduces the number of plans that we need to consider. However, it still takes too long to consider all policies and determine one of minimal expected cost. Thus, we now study more sophisticated search algorithms (here: stochastic dynamic programming algorithms).

11

Decision-Theoretic Planning

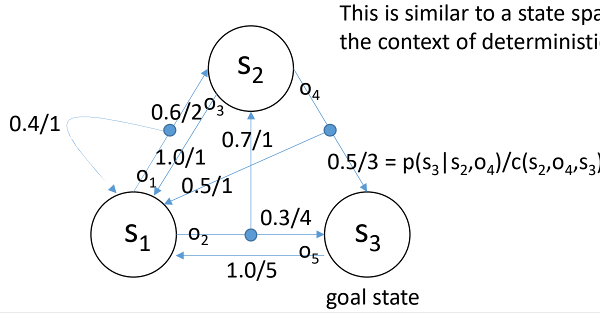
- We now study the case where we have a model available, that is, know all actions and their effects. This model is specified as an MDP (Markov Decision Process). We use this model for planning.

12

MDP Notation

- We do not need to label the start state since we will find a policy with minimal expected plan-execution cost from any state to the goal state.
- The stop action is automatically assigned to all goal states (here: s_3).

This is similar to a state space in the context of deterministic searches.



13

MDP Planning

Different from deterministic search, the policy with minimal expected plan-execution cost can have cycles, which complicates planning.

- We have a chicken-and-egg problem:
  - If one knows the optimal actions (o_2 in s_1, o_4 in s_2 and stop in s_3), one can calculate the expected goal distances as shown earlier:
    - c1 = 0.7 $(1+c_2)$ + 0.3 $(4+c_3)$ (= 5.08)
      c2 = 0.5 $(1+c_1)$ + 0.5 $(3+c_3)$ (= 4.54)
      c3 = 0

14

MDP Planning

Different from deterministic search, the policy with minimal expected plan-execution cost can have cycles, which complicates planning.

- We have a chicken-and-egg problem:
  - If one knows the expected goal distances (c_1=5.08 for s_1, c_2=4.54 for s_2 and c_3=0.00 for s_3), one can calculate the optimal actions by greedily assigning the action to each state that decreases the expected goal distance the most:
    - If one executes o_1 [o_2] in s_1 and uses the given expected goal distances as expected minimal costs to get from the resulting successor state to a goal state, then the total expected cost to get from s_1 to a goal state is 0.4 (1+c_1) + 0.6 (2+c_2) = 6.36 [0.7 (1+c_2) + 0.3 (4+c_3) = 5.08]. Since min(6.36,5.08) = 5.08, one should execute o_2 in s_1.
    - If one executes o_3 [o_4] in s_2 and uses the given expected goal distances as expected minimal costs to get from the resulting successor state to a goal state, then the total expected cost to get from s_2 to a goal state is 1.0 (1+c_1) = 6.08 [0.5 (1+c_1) + 0.5 (3+c_3) = 4.54]. Since min(6.08,4.54) = 4.54, one should execute o_4 in s_2.
    - One should stop in s_3 since s_3 is a goal state.

15

MDP Planning

- Unfortunately, one neither knows the optimal actions nor the expected goal distances. Thus, one needs to calculate them simultaneously. We present two methods for doing that, namely value iteration and policy iteration.

16

MDP Planning – Value Iteration

- In general, one solves the following system of equations oldEQ for calculating the expected plan-execution cost of policies:
  - $c_i = 0$ if s_i is a goal state
  - $c_i = \Sigma_k\, p(s_k|s_i,a(s_i))\, (c(s_i,a(s_i),s_k) + c_k)$ if s_i is not a goal state

Called Bellman equations after an ex-faculty member at USC!

- In general, one solves the following system of equations EQ for finding a policy with minimal expected plan-execution cost (where c_i is the expected goal distance of state s_i, that is, the expected plan-execution cost if one starts in state s_i and follows the optimal policy):
  - $c_i = 0$ if s_i is a goal state
  - $c_i = \min_{a \text{ executable in } s_i} \Sigma_k\, p(s_k|s_i,a)\, (c(s_i,a,s_k) + c_k)$ if s_i is not a goal state

17

MDP Planning – Value Iteration

- One solves the system of equations oldEQ as follows:
  - for all i
    - $c_{i,0} = 0$
  - for t=0 to ∞
    - for all i
      - $c_{i,t+1} = 0$ if s_i is a goal state
      - $c_{i,t+1} = \Sigma_k\, p(s_k|s_i,a(s_i))\, (c(s_i,a(s_i),s_k) + c_{k,t})$ if s_i is not a goal state

The typical termination criterion is: $|c_{i,t+1} - c_{i,t}| < \varepsilon$ for all i (for a given small positive ε).

- One solves the system of equations EQ as follows with value iteration:
  - for all i
    - $c_{i,0} = 0$
  - for t=0 to ∞
    - for all i
      - $c_{i,t+1} = 0$ if s_i is a goal state
      - $c_{i,t+1} = \min_{a \text{ executable in } s_i} \Sigma_k\, p(s_k|s_i,a)\, (c(s_i,a,s_k) + c_{k,t})$ if s_i is not a goal state

Pick values $c_{i,0}$

Improve values $c_{i,t}$ to values $c_{i,t+1}$ by calculating

18

MDP Planning – Value Iteration

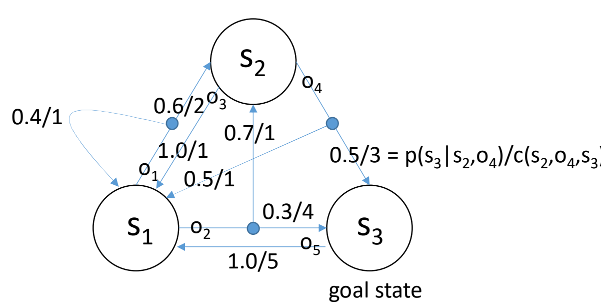
- $c_{i,t}$ = expected plan-execution cost if one starts in state s_i, follows the optimal policy, and stops in a goal state or after executing exactly t actions (whatever comes earlier)
- $c_i = \lim_{t\to\infty} c_{i,t}$ for all i
- If one is currently in state s_i and stops in goal states or after executing exactly t actions, then one should execute action "stop" if s_i is a goal state or t=0 and action
  $\text{argmin}_{a \text{ executable in } s_i} \Sigma_k\, p(s_k|s_i,a)\, (c(s_i,a,s_k) + c_{k,t-1})$
  otherwise.
- If one is currently in state s_i and stops in goal states, then one should execute action "stop" if s_i is a goal state and action
  $\text{argmin}_{a \text{ executable in } s_i} \Sigma_k\, p(s_k|s_i,a)\, (c(s_i,a,s_k) + c_k)$
  otherwise.

MDP Planning – Value Iteration

```
#include <stdio.h>

main()
{
  float s1=0.0, s2=0.0, s3=0.0;
  float o1,o2,o3,o4;
  char *ostar1, *ostar2;
  int i;

  for (i=1; i<10000; ++i)
    {
      o1 = 0.4*(1.0+s1)+0.6*(2.0+s2);
      o2 = 0.7*(1.0+s2)+0.3*(4.0+s3);
      o3 = 1.0*(1.0+s1);
      o4 = 0.5*(1.0+s1)+0.5*(3.0+s3);
      if (o1<o2)
        {
          s1 = o1;
          ostar1 = "o1";
        }
      else
        {
          s1 = o2;
          ostar1 = "o2";
        }
      if (o3<o4)
        {
          s2 = o3;
          ostar2 = "o3";
        }
      else
        {
          s2 = o4;
          ostar2 = "o4";
        }
      printf("iteration %5d: s1 (execute %s) %.2f (o1 %.2f, o2 %.2f), s2 (execute %s) %.2f (o3 %.2f, o4 %.2f), s3 (execute stop) %.2f\n",
             i, ostar1, s1, o1, o2, ostar2, s2, o3, o4, s3);
    }
}
```



MDP Planning – Value Iteration

$c_{1,t}$ $c_{2,t}$ $c_{3,t}$

```
iteration    1: s1 (execute o1) 1.60 (o1 1.60, o2 1.90), s2 (execute o3) 1.00 (o3 1.00, o4 2.00), s3 (execute stop) 0.00
iteration    2: s1 (execute o2) 2.60 (o1 2.84, o2 2.60), s2 (execute o3) 2.60 (o3 2.60, o4 2.80), s3 (execute stop) 0.00
iteration    3: s1 (execute o2) 3.72 (o1 4.20, o2 3.72), s2 (execute o4) 3.30 (o3 3.60, o4 3.30), s3 (execute stop) 0.00
iteration    4: s1 (execute o2) 4.21 (o1 5.07, o2 4.21), s2 (execute o4) 3.86 (o3 4.72, o4 3.86), s3 (execute stop) 0.00
iteration    5: s1 (execute o2) 4.60 (o1 5.60, o2 4.60), s2 (execute o4) 4.11 (o3 5.21, o4 4.11), s3 (execute stop) 0.00
iteration    6: s1 (execute o2) 4.77 (o1 5.90, o2 4.77), s2 (execute o4) 4.30 (o3 5.60, o4 4.30), s3 (execute stop) 0.00
iteration    7: s1 (execute o2) 4.91 (o1 6.09, o2 4.91), s2 (execute o4) 4.39 (o3 5.77, o4 4.39), s3 (execute stop) 0.00
iteration    8: s1 (execute o2) 4.97 (o1 6.20, o2 4.97), s2 (execute o4) 4.46 (o3 5.91, o4 4.46), s3 (execute stop) 0.00
iteration    9: s1 (execute o2) 5.02 (o1 6.26, o2 5.02), s2 (execute o4) 4.49 (o3 5.97, o4 4.49), s3 (execute stop) 0.00
iteration   10: s1 (execute o2) 5.04 (o1 6.30, o2 5.04), s2 (execute o4) 4.51 (o3 6.02, o4 4.51), s3 (execute stop) 0.00
...
iteration 9995: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
iteration 9996: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
iteration 9997: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
iteration 9998: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
iteration 9999: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
```

21

MDP Planning – Value Iteration

```
iteration t= 1: s1 (execute o1) 1.60 (o1 1.60, o2 1.90), s2 (execute o3) 1.00 (o3 1.00, o4 2.00), s3 (execute stop) 0.00
iteration t= 2: s1 (execute o2) 2.60 (o1 2.84, o2 2.60), s2 (execute o3) 2.60 (o3 2.60, o4 2.80), s3 (execute stop) 0.00
iteration t= 3: s1 (execute o2) 3.72 (o1 4.20, o2 3.72), s2 (execute o4) 3.30 (o3 3.60, o4 3.30), s3 (execute stop) 0.00
iteration t= 4: s1 (execute o2) 4.21 (o1 5.07, o2 4.21), s2 (execute o4) 3.86 (o3 4.72, o4 3.86), s3 (execute stop) 0.00
iteration t= 5: s1 (execute o2) 4.60 (o1 5.60, o2 4.60), s2 (execute o4) 4.11 (o3 5.21, o4 4.11), s3 (execute stop) 0.00
iteration t= 6: s1 (execute o2) 4.77 (o1 5.90, o2 4.77), s2 (execute o4) 4.30 (o3 5.60, o4 4.30), s3 (execute stop) 0.00
iteration t= 7: s1 (execute o2) 4.91 (o1 6.09, o2 4.91), s2 (execute o4) 4.39 (o3 5.77, o4 4.39), s3 (execute stop) 0.00
iteration t= 8: s1 (execute o2) 4.97 (o1 6.20, o2 4.97), s2 (execute o4) 4.46 (o3 5.91, o4 4.46), s3 (execute stop) 0.00
iteration t= 9: s1 (execute o2) 5.02 (o1 6.26, o2 5.02), s2 (execute o4) 4.49 (o3 5.97, o4 4.49), s3 (execute stop) 0.00
iteration t=10: s1 (execute o2) 5.04 (o1 6.30, o2 5.04), s2 (execute o4) 4.51 (o3 6.02, o4 4.51), s3 (execute stop) 0.00
...
iteration 9995: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
iteration 9996: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
iteration 9997: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
iteration 9998: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
iteration 9999: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
```

c_1 c_2 c_3

- If one can stop after executing exactly 3 actions, then
  - first action execution (t=3): execute o_2 in s_1, o_4 in s_2 and stop in s_3 (see iteration 3)
  - second action execution (t=2): execute o_2 in s_1, o_3 in s_2 and stop in s_3 (see iteration 2)
  - third action execution (t=1): execute o_1 in s_1, o_3 in s_2 and stop in s_3 (see iteration 1)
- This is not a policy!

22

MDP Planning – Value Iteration

```
iteration    1: s1 (execute o1) 1.60 (o1 1.60, o2 1.90), s2 (execute o3) 1.00 (o3 1.00, o4 2.00), s3 (execute stop) 0.00
iteration    2: s1 (execute o2) 2.60 (o1 2.84, o2 2.60), s2 (execute o3) 2.60 (o3 2.60, o4 2.80), s3 (execute stop) 0.00
iteration    3: s1 (execute o2) 3.72 (o1 4.20, o2 3.72), s2 (execute o4) 3.30 (o3 3.60, o4 3.30), s3 (execute stop) 0.00
iteration    4: s1 (execute o2) 4.21 (o1 5.07, o2 4.21), s2 (execute o4) 3.86 (o3 4.72, o4 3.86), s3 (execute stop) 0.00
iteration    5: s1 (execute o2) 4.60 (o1 5.60, o2 4.60), s2 (execute o4) 4.11 (o3 5.21, o4 4.11), s3 (execute stop) 0.00
iteration    6: s1 (execute o2) 4.77 (o1 5.90, o2 4.77), s2 (execute o4) 4.30 (o3 5.60, o4 4.30), s3 (execute stop) 0.00
iteration    7: s1 (execute o2) 4.91 (o1 6.09, o2 4.91), s2 (execute o4) 4.39 (o3 5.77, o4 4.39), s3 (execute stop) 0.00
iteration    8: s1 (execute o2) 4.97 (o1 6.20, o2 4.97), s2 (execute o4) 4.46 (o3 5.91, o4 4.46), s3 (execute stop) 0.00
iteration    9: s1 (execute o2) 5.02 (o1 6.26, o2 5.02), s2 (execute o4) 4.49 (o3 5.97, o4 4.49), s3 (execute stop) 0.00
iteration   10: s1 (execute o2) 5.04 (o1 6.30, o2 5.04), s2 (execute o4) 4.51 (o3 6.02, o4 4.51), s3 (execute stop) 0.00
...
iteration 9995: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
iteration 9996: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
iteration 9997: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
iteration 9998: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
iteration 9999: s1 (execute o2) 5.08 (o1 6.35, o2 5.08), s2 (execute o4) 4.54 (o3 6.08, o4 4.54), s3 (execute stop) 0.00
```

- If one has to execute actions forever, then
  - always ($t=\infty$): execute o_2 in s_1, o_4 in s_2 and stop in s_3 (see iteration 9999)
- This is a policy!

23

MDP Planning – Policy Iteration

- One solves the system of equations EQ as follows with policy iteration:

Pick policy $a_0(s_i)$

- for all i
  - Pick an $a_0(s_i)$ from all actions executable in s_i so that a goal state can be reached from every state with positive probability
- for n=0 to ∞ — The typical termination criterion is: $a_{n+1}(s_i) = a_n(s_i)$ for all i.

Evaluate policy $a_n(s_i)$ by calculating the c_i

  - for all i
    - $c_{n,i,0} = 0$
  - for t=0 to ∞ — The typical termination criterion is: $|c_{n,i,t+1} - c_{n,i,t}| < \varepsilon$ for all i (for a given small positive ε
    - for all i
      - $c_{n,i,t+1} = 0$ — if s_i is a goal state
      - $c_{n,i,t+1} = \Sigma_k\, p(s_k|s_i,a_n(s_i))\, (c(s_i,a_n(s_i),s_k) + c_{n,k,t})$ — if s_i is not a goal state

Improve policy $a_n(s_i)$ to policy $a_{n+1}(s_i)$

  - for all i
    - $c_{n,i} = \lim_{t\to\infty} c_{n,i,t}$
  - for all i — Use $a_{n+1}(s_i) = a_n(s_i)$ if $a_n(s_i)$ is still optimal.
    - $a_{n+1}(s_i) = \text{stop}$ — if s_i is a goal state
    - $a_{n+1}(s_i) = \text{argmin}_{a \text{ executable in } s_i} \Sigma_k\, p(s_k|s_i,a)\, (c(s_i,a,s_k) + c_{n,k})$ — if s_i is not a goal state

24

MDP Planning – Policy Iteration

- If one is currently in state s_i and stops in goal states, then one should execute action $a_n(s_i)$ in state s_i, where n is the largest iteration.

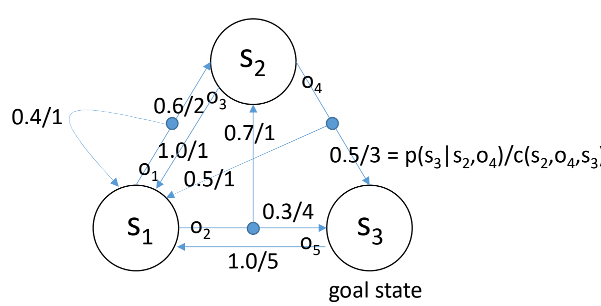
25

MDP Planning – Policy Iteration

```
main(){
    int a1=2, a2=3;
    int olda1, olda2;
    float s1=0.0, s2=0.0, s3=0.0;
    float o1, o2, o3, o4;
    int i=0, j;
    const char *name[4];

    name[0] = "o1";
    name[1] = "o2";
    name[2] = "o3";
    name[3] = "o4";

    do{
        olda1 = a1;
        olda2 = a2;
        for(j=1; j<10000; j++){
            o1 = 0.4*(1.0+s1) + 0.6*(2.0+s2);
            o2 = 0.7*(1.0+s2) + 0.3*(4.0+s3);
            o3 = 1.0*(1.0+s1);
            o4 = 0.5*(1.0+s1) + 0.5*(3.0+s3);
            if(a1==1)
                s1 = o1;
            else
                s1 = o2;
            if(a2==3)
                s2 = o3;
            else
                s2 = o4;
        }
        if(o1<o2)
            a1 = 1;
        else
            a1 = 2;
        if(o3<o4)
            a2 = 3;
        else
            a2 = 4;
        printf("iteration %5d: s1 (execute %s) %5.2f (o1 %5.2f, o2 %5.2f), s2 (execute %s) %5.2f (o3 %5.2f, o4 %5.2f), s3 (execute stop) %5.2f \n",
                i++, name[a1-1], s1, o1, o2, name[a2-1], s2, o3, o4, s3);
    }
    while(a1!=olda1 || a2!=olda2);
}
```



26

MDP Planning – Policy Iteration

$a_{iteration}(s_1)$ $a_{iteration}(s_2)$ $a_{iteration}(s_3)$

```
iteration     0: s1 (execute o2)  8.67 (o1 10.87, o2  8.67), s2 (execute o4)  9.67 (o3  9.67, o4  6.33), s3 (execute stop)  0.00
iteration     1: s1 (execute o2)  5.08 (o1  6.35, o2  5.08), s2 (execute o4)  4.54 (o3  6.08, o4  4.54), s3 (execute stop)  0.00
```

27

MDP Planning – Policy Iteration

$a_{iteration}(s_1)$ $a_{iteration}(s_2)$ $a_{iteration}(s_3)$

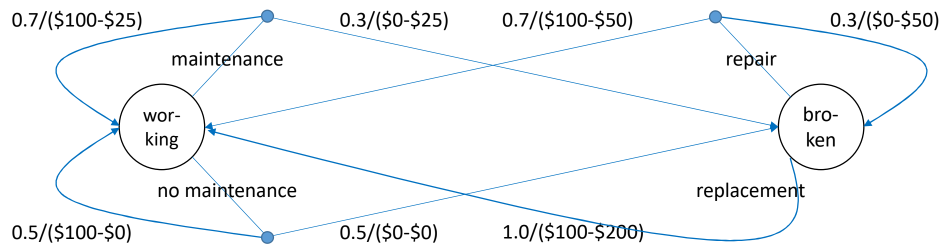
```
iteration     0: s1 (execute o2)  8.67 (o1 10.87, o2  8.67), s2 (execute o4)  9.67 (o3  9.67, o4  6.33), s3 (execute stop)  0.00
iteration     1: s1 (execute o2)  5.08 (o1  6.35, o2  5.08), s2 (execute o4)  4.54 (o3  6.08, o4  4.54), s3 (execute stop)  0.00
```

- If one has to execute actions forever, then
  - always: execute o_2 in s_1, o_4 in s_2 and stop in s_3 (see iteration 1)
- This is a policy!

28

MDP Planning with Discounting

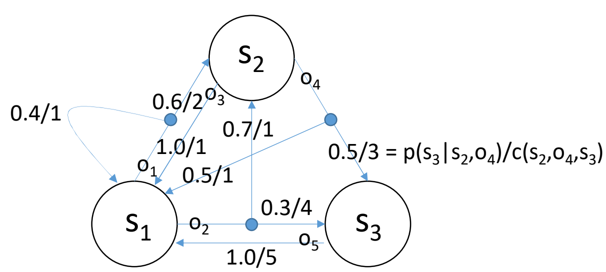
- What if there is no goal state, that is, one has to execute actions forever?
- Consider maintenance planning for a machine that is either working or broken.



29

MDP Planning with Discounting

- What if there is no goal state, that is, one has to execute actions forever?



- Every policy now has expected plan-execution cost infinity, that is, looks equally good.
- Infinite plan-execution costs cause problems, since it is preferable to incur, say, the infinite sequence of action costs, 1 1 1 1 1 1 … than 5 5 5 5 5 5 …!

30

MDP Planning with Discounting

- One needs to change the planning objective, e.g. to
  - minimizing the expected plan execution cost per action execution or
  - minimizing the expected discounted plan execution cost.
- Minimizing the expected discounted plan execution cost is a bit simpler, so we will do that in the following.
- Everything that we do in the following can be (and is) also done if there are goal states even though it is not necessary.

31

MDP Planning with Discounting

America's Got Talent Winner is Not an Instant Millionaire

Last night, NBC's *America's Got Talent* announced the winner of its sixth season. Landau Eugene Murphy, Jr., a 36 year old car wash detailer from West Virginia, was overcome with emotion as he was told of the $1 million prize and the opportunity to headline a show at Caesar's Palace in Las Vegas.

...

But if you read the fine print on the screen at the end of the finale last night, the million dollar prize is actually a 40-year long annuity. In reality, Murphy, whose impressive singing voice resembles that of Frank Sinatra, can expect an annual payout of only $25,000—before taxes, that is. Murphy will be offered a lump cash payment in lieu of the annuity, but this will likely be in the $300,000 range (again, before taxes).

Source: Forbes, September 15, 2011

32

MDP Planning with Discounting

- A similar example with fewer payouts (to better fit on the slide):

Jan 1, 2012	Jan 1, 2013	Jan 1, 2014	Jan 1, 2015
\$25,000	\$25,000	\$25,000	\$25,000

33

MDP Planning with Discounting

- If we put \$1 into a savings account with interest rate p%, then we have \$(1 + p/100) in the savings account after one year.

Jan 1, 2012	Jan 1, 2013
\$1 $\xrightarrow{\cdot (100+p)/100}$ $\xleftarrow{\cdot 100/(100+p)}$	\$(1 + p/100)

- We call $0 < 100/(100+p) \leq 1$ the discount factor γ (gamma).

34

MDP Planning with Discounting

- A similar example with fewer payouts (to better fit on the slide):

Jan 1, 2012	Jan 1, 2013	Jan 1, 2014	Jan 1, 2015
\$25,000	\$25,000	\$25,000	\$25,000

$$\begin{array}{cccccc} + \gamma\,(1+\gamma+\gamma^2)\ \$25{,}000 & \cdot\gamma & + \gamma\,(1+\gamma)\ \$25{,}000 & \cdot\gamma & + \gamma\ \$25{,}000 & \cdot\gamma \\ \hline (1+\gamma+\gamma^2+\gamma^3)\ \$25{,}000 & & (1+\gamma+\gamma^2)\ \$25{,}000 & & (1+\gamma)\ \$25{,}000 & \end{array}$$

- So, for an interest rate of 5% (that is, a discount factor of $\gamma \approx 0.952$), providing an annuity of 4 payments of \$25,000 each year and a lump sum payoff of $(1+\gamma+\gamma^2+\gamma^3)$ \$25,000 ≈ \$93,081.20 (called the total discounted cost of the annuity) are equally preferable.

35

MDP Planning with Discounting

- Assume that the discount factor γ is 0.95 and one wants to minimize the expected discounted plan-execution cost.
- The infinite sequence of action costs 1 1 1 1 1 … has a finite(!) discounted plan-execution cost of $(1+\gamma+\gamma^2+\ldots)\ 1 = 1/(1-\gamma)\ 1 = 20$.
- The infinite sequence of action costs 5 5 5 5 5 … has a finite(!) discounted plan-execution cost of $(1+\gamma+\gamma^2+\ldots)\ 5 = 1/(1-\gamma)\ 5 = 100$.
- So, one now prefers the infinite sequence of action costs 1 1 1 1 1 … over the infinite sequence of action costs 5 5 5 5 5 …!

36

MDP Planning with Discounting

- A similar example with fewer payouts (to better fit on the slide):

Jan 1, 2012	
\$25,000	cost at time 2012 (t)
+ γ $(1+ \gamma+ \gamma^2)$ \$25,000	+ γ expected discounted plan-execution cost at time 2013 (t+1) [from time 2013 on]
$(1+ \gamma+ \gamma^2+\gamma^3)$ \$25,000	expected discounted plan-execution cost at time 2012 (t) [from time 2012 on]

Earlier, we used

cost at time 2012 (t)
+ expected plan-execution cost at time 2013 (t+1) [from time 2013 on]
expected plan-execution cost at time 2012 (t) [from time 2012 on]

MDP Planning with Discounting – Value Iteration

- In general, one solves the following system of equations oldEQ' for calculating the expected discounted plan-execution cost of policies:
  - $c_i = 0$ if s_i is a goal state
  - $c_i = \Sigma_k\, p(s_k|s_i,a(s_i))\, (c(s_i,a(s_i),s_k) + \gamma\, c_k)$ if s_i is not a goal state

- In general, one solves the following system of equations EQ' for finding a policy with minimal expected discounted plan-execution cost (where c_i is the expected discounted plan-execution cost if one starts in state s_i and follows the optimal policy):
  - $c_i = 0$ if s_i is a goal state
  - $c_i = \min_{a \text{ executable in } s_i} \Sigma_k\, p(s_k|s_i,a)\, (c(s_i,a,s_k) + \gamma\, c_k)$ if s_i is not a goal state

MDP Planning with Discounting – Value Iteration

- One solves the system of equations oldEQ' as follows:
  - for all i
    - $c_{i,0} = 0$
  - for t=0 to ∞
    - for all i
      - $c_{i,t+1} = 0$ if s_i is a goal state
      - $c_{i,t+1} = \Sigma_k\ p(s_k|s_i,a(s_i))\ (c(s_i,a(s_i),s_k) + \gamma\ c_{k,t})$ if s_i is not a goal state

The typical termination criterion is: $|c_{i,t+1} - c_{i,t}| < \varepsilon$ for all i (for a given small positive ε).

- One solves the system of equations EQ' as follows with value iteration:
  - for all i
    - $c_{i,0} = 0$
  - for t=0 to ∞
    - for all i
      - $c_{i,t+1} = 0$ if s_i is a goal state
      - $c_{i,t+1} = \min_{a \text{ executable in } s_i} \Sigma_k\ p(s_k|s_i,a)\ (c(s_i,a,s_k) + \gamma\ c_{k,t})$ if s_i is not a goal state

Pick values $c_{i,0}$

Improve values $c_{i,t}$ to values $c_{i,t+1}$ by calculating

39

MDP Planning with Discounting – Value Iteration

- $c_{i,t}$ = expected discounted plan-execution cost if one starts in state s_i, follows the optimal policy, and stops in a goal state or after executing exactly t actions (whatever comes earlier).
- It holds that $c_i = \lim_{t\to\infty} c_{i,t}$ for all i.
- If one is currently in state s_i and stops in goal states or after executing exactly t actions, then one should execute action "stop" if s_i is a goal state or t=0 and action $\text{argmin}_{a \text{ executable in } s_i} \Sigma_k\ p(s_k|s_i,a)\ (c(s_i,a,s_k) + \gamma\ c_{k,t-1})$ otherwise.
- If one is currently in state s_i and stops in goal states, then one should execute action "stop" if s_i is a goal state and action $\text{argmin}_{a \text{ executable in } s_i} \Sigma_k\ p(s_k|s_i,a)\ (c(s_i,a,s_k) + \gamma\ c_k)$ otherwise.

40

MDP Planning with Discounting – Value Iteration

- The policy with minimal expected discounted plan-execution cost depends on the discount factor.
- The discount factor cannot be 1 since this corresponds to finding a policy with minimal expected plan-execution cost but, ideally, it should be close to 1 (e.g. 0.95 or 0.99).
- The smaller it is, the higher one weighs costs incurred in the immediate future over costs incurred in the distance future.
- The discount factor can also be interpreted as the probability of not dying.
  - If the interest rate is $(1-\gamma)/\gamma$,
    then the value of receiving x dollars in a year is γ x dollars right now.
  - If I die later this year with probability $1-\gamma$ and can thus no longer receive future payoffs, then the expected value of receiving x dollars in a year is γ x dollars right now

41

MDP Planning with Discounting – Policy Iteration

- One solves the system of equations EQ' as follows with policy iteration:

Pick policy $a_0(s_i)$

- for all i
  - Pick an $a_0(s_i)$ from all actions executable in s_i so that a goal state can be reached from every state with positive probability
- for n=0 to ∞ — The typical termination criterion is: $a_{n+1}(s_i) = a_n(s_i)$ for all i.

Evaluate policy $a_n(s_i)$ by calculating the c_i

  - for all i
    - $c_{n,i,0} = 0$
  - for t=0 to ∞ — The typical termination criterion is: $|c_{n,i,t+1} - c_{n,i,t}| < \varepsilon$ for all i (for a given small positive ε)
    - for all i
      - $c_{n,i,t+1} = 0$ if s_i is a goal state
      - $c_{n,i,t+1} = \Sigma_k\, p(s_k|s_i,a_n(s_i))\, (c(s_i,a_n(s_i),s_k) + \gamma\, c_{n,k,t})$ if s_i is not a goal state

Improve policy $a_n(s_i)$ to policy $a_{n+1}(s_i)$

  - for all i
    - $c_{n,i} = \lim_{t\to\infty} c_{n,i,t}$
  - for all i — Use $a_{n+1}(s_i) = a_n(s_i)$ if $a_n(s_i)$ is still optimal.
    - $a_{n+1}(s_i) = \text{stop}$ if s_i is a goal state
    - $a_{n+1}(s_i) = \text{argmin}_{a \text{ executable in } s_i}\, \Sigma_k\, p(s_k|s_i,a)\, (c(s_i,a,s_k) + \gamma\, c_{n,k})$ if s_i is not a goal state

42

MDP Planning with Discounting – Policy Iteration

- If one is currently in state s_i and stops in goal states, then one should execute action $a_n(s_i)$ in state s_i, where n is the largest iteration.

43

Decision-Theoretic Planning

- We now study the case where we do not have a model available, that is, do not know all actions and their effects. We only know which state the agent is currently in and which actions it has available. We thus cannot plan but we can still use reinforcement learning (RL) to learn which action the agent should choose in its current state.

44

MDP Planning with Discounting – Value Iteration

- One solves the system of equations as follows with value iteration:
  - for all i
    - $c_{i,0} = 0$
  - for t=0 to ∞
    - for all i
      - $c_{i,t+1} = 0$ if s_i is a goal state
      - $c_{i,t+1} = \min_{a \text{ executable in } s_i} \underbrace{\Sigma_k\, p(s_k | s_i,a)\, (c(s_i,a,s_k) + \gamma\, c_{k,t})}_{\text{q-value } q_{t+1}(s_i,a)}$ if s_i is not a goal state

RL with Discounting – Q Learning

- The agent executes
  - for all states s and actions a
    - $q(s,a) = 0$
  - s = start state of the agent
  - repeat
    - $a = \begin{cases} \text{random action executable in s with probability } \varepsilon \\ \text{argmin}_{a \text{ executable in s}}\, q(s,a) \text{ with probability } 1-\varepsilon \end{cases}$
    - execute a and observe the resulting action cost c and successor state s’
    - $q(s,a) = q(s,a) + \alpha\, (c + \gamma \underbrace{\min_{a' \text{ executable in } s'} q(s',a')}_{\text{If } s'=s_i\text{, then this is an estimate of } c_i} - q(s,a))$
    - s = s’
  - until s is a goal state

RL with Discounting – Q Learning

- The agent executes
  - for all states s and actions a
    - q(s,a) = 0
  - s = start state of the agent
  - repeat
    - a = { random action executable in s with probability ε ; $\text{argmin}_{a \text{ executable in } s}$ q(s,a) with probability 1-ε }
    - execute a and observe the resulting action cost c and successor state s'
    - q(s,a) = q(s,a) + α (c + γ $\underbrace{\min_{a' \text{ executable in } s'} q(s',a')}_{\text{If } s'=s_i \text{, then this is an estimate of } c_i}$ – q(s,a))
    - s = s'
  - until s is a goal state

From time to time, the agent needs to execute seemingly suboptimal actions to explore the executable actions and potentially discover actions that are better than the currently seemingly best action. Thus, it needs an exploration policy. The one used here is called ε-greedy.

Exploration (here: execute a random action)

Exploitation (execute the seemingly best action

This should look familiar from gradient descent

47

RL with Discounting – Q Learning

- Q(s,a) is called the q-value of action a in state s. It is an estimate of the total expected discounted plan execution cost when executing action a in state s and then following the optimal policy (until a goal state is reached, if there is one). The agent should thus always execute the action in its current state with the smallest q-value.
- RL often uses rewards instead of costs, where a reward is just a negative cost. In this case, Q learning needs to maximize instead of minimize.
- The learning rate 0 < α is often close to zero, the exploration probability 0 < ε is often close to zero, and the discount factor 0 < γ < 1 is often close to one.

48