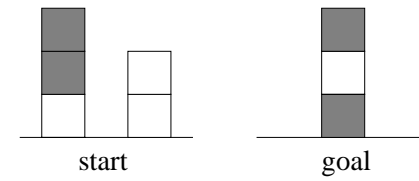


Artificial Intelligence

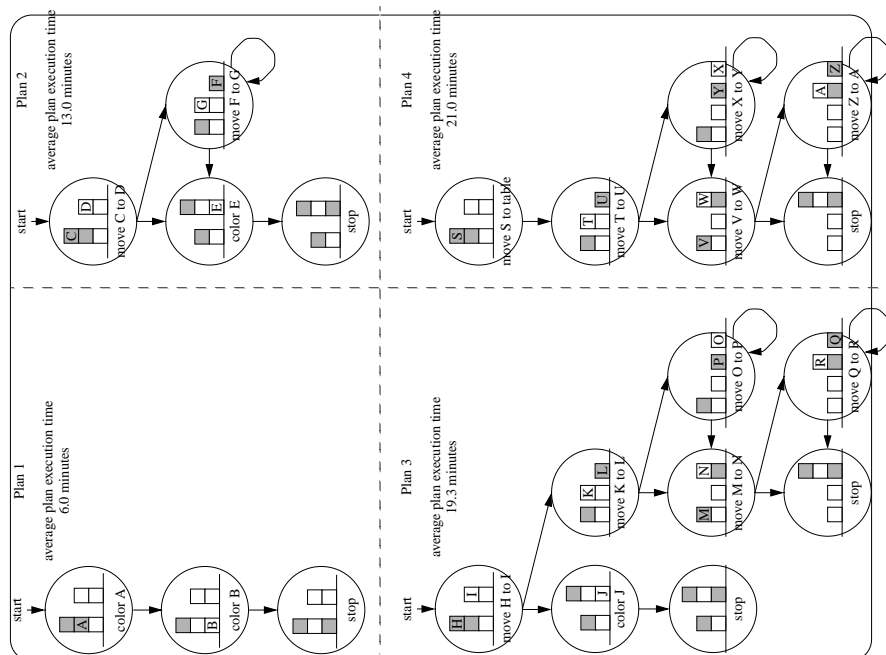
Markov Decision Problems

Nilsson - briefly mentioned in Chapter 10
 Russell and Norvig - Chapters 17 and 20

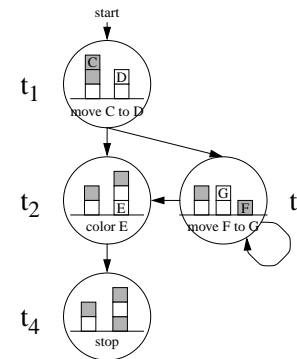
example: a probabilistic blocksworld



action	outcome	probability	time (= cost)
move	success	0.1	1 minute
	failure	0.9	1 minute
paint	success	1.0	3 minutes



how to determine the average plan-execution time of a given plan

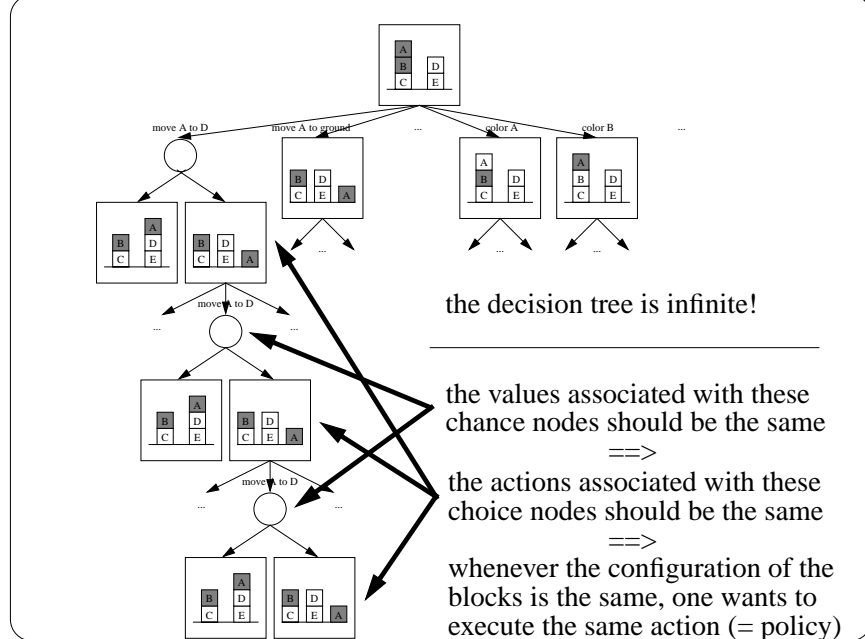


t_i = average plan-execution time until a goal state is reached if the agent starts in state i and follows the plan

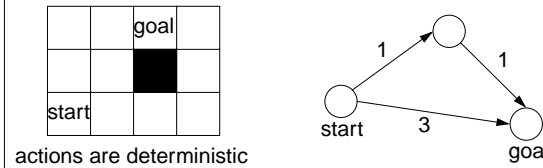
$$\begin{aligned}
 t_1 &= 1 + 0.1 t_2 + 0.9 t_3 \\
 t_2 &= 3 + 1.0 t_4 \\
 t_3 &= 1 + 0.1 t_2 + 0.9 t_3 \\
 t_4 &= 0
 \end{aligned}$$

$$\begin{aligned}
 t_1 &= 13 \quad (= \text{average plan-execution time}) \\
 t_2 &= 3 \\
 t_3 &= 13 \\
 t_4 &= 0
 \end{aligned}$$

how to (almost) determine a plan with minimal average plan-execution time



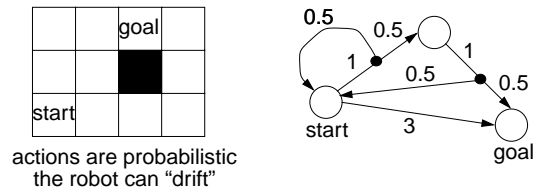
deterministic planning and search



actions are deterministic

actions have deterministic effects
 state and action determine uniquely the successor state
 states are completely observable
 a plan is a sequence of actions (= path)
 minimize total cost
 optimal plan is acyclic

probabilistic planning and search
 =Markov Decision Problems (MDPs)



actions are probabilistic
 the robot can "drift"

Markov property

actions have probabilistic effects
 state and action uniquely determine prob distribution over successor states
 states are completely observable
 a plan is a mapping from states to actions (= policy)
 minimize expected total cost
 optimal plan can be cyclic

deterministic planning and search



N N E E

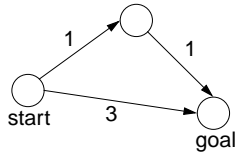
a plan is a sequence of actions (= path)
 can be found using (forward or backward) search

Markov Decision Problems

a plan is a mapping from states to actions (= policy)
 how to find it?

determine the expected goal distances of all states
 ↓
 greedily assign the action to each state
 that decreases the expected goal distance the most

deterministic planning and search



s = state
 a = action
 $A(s)$ = set of actions that can be executed in state s
 $\text{succ}(s,a)$ = the state that results from the execution of action a in state s
 $c(s,a)$ = the cost that results from the execution of action a in state s

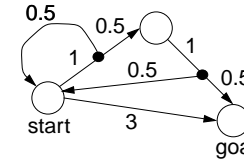
$gd(s)$ = goal distance of state s

$gd(s) = 0$ if s is a goal state
 $gd(s) = \min_{a \in A(s)} c(s,a) + gd(\text{succ}(s,a))$ if s is not a goal state

$a(s)$ = the optimal action to execute in state s

$a(s) = \text{argmin}_{a \in A(s)} c(s,a) + gd(\text{succ}(s,a))$ if s is not a goal state

Markov Decision Problems



s = state
 a = action
 $A(s)$ = set of actions that can be executed in state s
 $\text{succ}(s,a)$ = the set of states that can result from the execution of action a in state s
 $c(s,a)$ = the cost that results from the execution of action a in state s
 $p(s'|s,a)$ = the probability that state s' results from the execution of action a in state s
 $gd(s)$ = expected goal distance of state s

Bellman equation

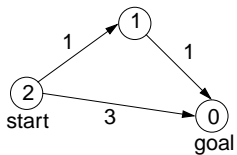
$gd(s) = 0$ if s is a goal state
 $gd(s) = \min_{a \in A(s)} (c(s,a) + \sum_{s' \in \text{succ}(s,a)} p(s'|s,a) gd(s'))$ if s is not a goal state

$a(s)$ = the optimal action to execute in state s

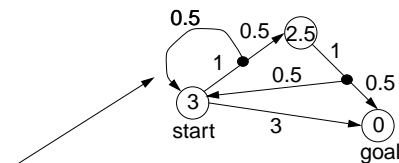
$a(s) = \text{argmin}_{a \in A(s)} (c(s,a) + \sum_{s' \in \text{succ}(s,a)} p(s'|s,a) gd(s'))$ if s is not a goal state

examples

deterministic planning and search

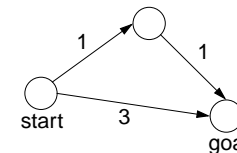


Markov Decision Problems



given the expected goal distances, we can use the definition to check them but calculating them is a chicken-and-egg problem

deterministic planning and search



s = state
 a = action
 $A(s)$ = set of actions that can be executed in state s
 $\text{succ}(s,a)$ = the state that results from the execution of action a in state s
 $c(s,a)$ = the cost that results from the execution of action a in state s

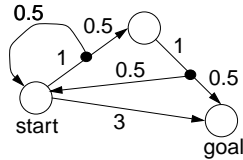
$gd(s)$ = goal distance of state s (= minimal cost until a goal is reached if execution starts in state s)
 $gd_i(s)$ = minimal cost until a goal is reached or i actions have been executed if execution starts in state s

for i larger than a constant: $gd(s) = gd_i(s)$ (= once $gd_i(s) = gd_{i-1}(s)$ for all states s)

$gd_0(s) = 0$

$gd_i(s) = 0$ if s is a goal state
 $gd_i(s) = \min_{a \in A(s)} (c(s,a) + gd_{i-1}(\text{succ}(s,a)))$ if s is not a goal state

Markov Decision Problems



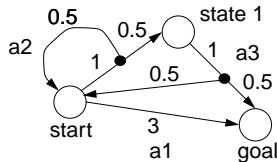
- s = state
- a = action
- A(s) = set of actions that can be executed in state s
- succ(s,a) = the set of states that can result from the execution of action a in state s
- c(s,a) = the cost that results from the execution of action a in state s
- p(s'|s,a) = the probability that state s' results from the execution of action a in state s
- gd(s) = expected goal distance of state s
- gd_i(s) = minimal expected cost until a goal is reached or i actions have been executed if execution starts in state s
- gd(s) = lim_{i → infinity} gd_i(s) (not necessarily after a finite amount of time)
- gd₀(s) = 0
- gd_i(s) = 0 if s is a goal state
- gd_i(s) = min_{a ∈ A(s)} (c(s,a) + Σ_{s' ∈ succ(s,a)} p(s'|s,a) gd_{i-1}(s')) if s is not a goal state

Value Iteration

maintains approximations of the goal distances (= values)

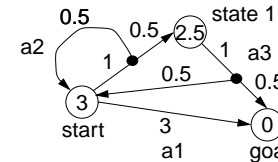
1. i := 0
2. Set (for all s ∈ S) gd_i(s) = 0.
3. i := i+1
4. Set (for all s ∈ S)
 - gd_i(s) = 0 if s is a goal state
 - gd_i(s) = min_{a ∈ A(s)} (c(s,a) + Σ_{s' ∈ succ(s,a)} p(s'|s,a) gd_{i-1}(s')) if s is not a goal state
5. If (for some s ∈ S) |gd_i(s) - gd_{i-1}(s)| > small constant, go to 3.
6. Set (for all s ∈ S that are not goal states)
 - a(s) = argmin_{a ∈ A(s)} (c(s,a) + Σ_{s' ∈ succ(s,a)} p(s'|s,a) gd_i(s'))

example of Value Iteration (1)



i		0	1	2	3	4	5	6
start	a1	0	3	3	3	3	3	3
	a2	0	1	2	2.75	3	3.375	3.75
state 1	a3	0	1	1.5	2	2	2.375	2.375
	a1	0	1	1.5	2	2	2.375	2.375
goal		0	0	0	0	0	0	0

example of Value Iteration (2)



(no discounting)

which action to execute in the start state?

- a1: 3 + 0 = 3
 - a2: 1 + 0.5 * 3 + 0.5 * 2.5 = 3.75
- execute a1!

Policy Iteration

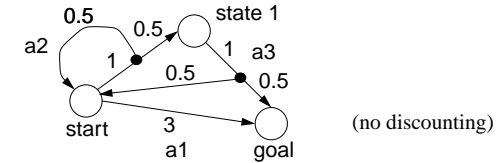
maintains a policy

1. $i := 0$
2. Set (for all $s \in S$ that are not goal states) $a_i(s)$ to an arbitrary action in $A(s)$.
3. Set $gd_i(s)$ to the average plan-execution time until a goal state is reached if the agent starts in state s and follows policy a_i
4. $i := i+1$
5. Set (for all $s \in S$ that are not goal states)

$$a_i(s) = \operatorname{argmin}_{a \in A(s)} (c(s,a) + \sum_{s' \in \operatorname{succ}(s,a)} p(s'|s,a) gd_i(s'))$$
6. If (for some $s \in S$ that is not a goal state) $a_i(s)$ does not equal $a_{i-1}(s)$, go to 3
7. Set (for all $s \in S$ that are not goal states) $a(s) = a_i(s)$.

Note: The initial policy a_0 has to guarantee that the agent reaches a goal state with probability one no matter which state it is started in.

example of Policy Iteration



policy at $i=0$
 $a_0(\text{start}) = a_2$ (could also have been a_1) and $a_0(\text{state } 1) = a_3$

$$gd_0(\text{start}) = 1 + 0.5 gd_0(\text{start}) + 0.5 gd_0(\text{state } 1) = 6$$

$$gd_0(\text{state } 1) = 1 + 0.5 gd_0(\text{start}) + 0.5 gd_0(\text{goal}) = 4$$

$$gd_0(\text{goal}) = 0$$

policy at $i=1$
 $a_1(\text{start}) = a_1$ and $a_1(\text{state } 1) = a_3$

$$gd_1(\text{start}) = 3 + 1.0 gd_0(\text{goal}) = 3$$

$$gd_1(\text{state } 1) = 1 + 0.5 gd_1(\text{start}) + 0.5 gd_1(\text{goal}) = 2.5$$

$$gd_1(\text{goal}) = 0$$

policy at $i=2$
 $a_2(\text{start}) = a_1$ and $a_2(\text{state } 1) = a_3$

execute action a_1 in the start state!

extensions: no goal (1)

what if there is no goal?
 "living in the world"

can no longer minimize expected cost until the goal is reached

- here:
- can minimize expected cost per action execution
 - can minimize expected total discounted cost

extensions: no goal (2)

cannot minimize expected total cost

1	2	3	4	4	4	4	...	expected total cost = infinite
1	1	1	1	1	1	1	...	expected total cost = infinite

extensions: no goal (3)

discount factor

total discounted cost =



if the interest rate is $(1-\gamma)/\gamma$ (for $0 < \gamma < 1$),
 how much money do I need to pay someone right now
 so that there is no difference to paying the following yearly installments

1 2 3 4 4 4 4 ...

x dollars right now are worth $(1 + (1-\gamma)/\gamma)x = x/\gamma$ dollars in a year
 so, y dollars in a year are worth γy dollars right now

answer: $1 + \gamma 2 + \gamma^2 3 + \gamma^3 4 + \gamma^4 4 + \dots$

extensions: no goal (4)

can minimize the expected total discounted cost - assume $\gamma = 0.9$

1 2 3 4 4 ... expected total discounted cost = 34.39
 1 1 1 1 1 ... expected total discounted cost = 10.00

extensions: no goal (5)

- discounting makes the total cost finite

c c c c c ... expected total discounted cost = $c/(1-\gamma)$

- discounting smoothes out the horizon

- discounting can be interpreted as the probability of dying

discounting:

if the interest rate is $(1-\gamma)/\gamma$,
 then y dollars in a year are worth γy dollars right now

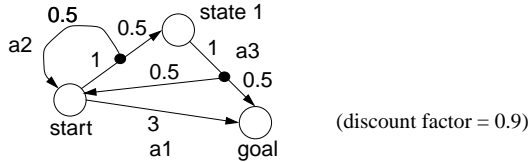
dying:

if I die later this year with probability $1-\gamma$,
 then the expected value of y dollars in a year is γy right now

Value-Iteration with or without discounting

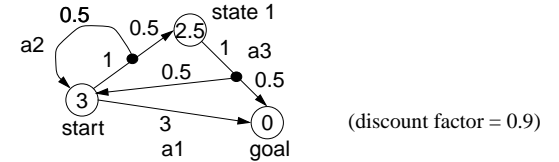
- γ = discount factor ($0 < \gamma < 1$); if there is a goal, can set $\gamma = 1$ (no discounting)
- s = state
- a = action
- $A(s)$ = set of actions that can be executed in state s
- $\text{succ}(s,a)$ = the set of states that can result from the execution of action a in state s
- $c(s,a)$ = the cost that results from the execution of action a in state s
- $p(s'|s,a)$ = the probability that state s' results from the execution of action a in state s
- $gd(s)$ = minimal expected discounted total cost if execution starts in state s
- $gd_0(s) = 0$ if s is a goal state
- $gd(s) = \min_{a \in A(s)} (c(s,a) + \gamma \sum_{s' \in \text{succ}(s,a)} p(s'|s,a) gd(s'))$ if s is not a goal state
- $gd_i(s)$ = minimal expected discounted total cost until a goal is reached or i actions have been executed if execution starts in state s
- $gd_0(s) = 0$ for all s
- $gd_i(s) = 0$ if s is a goal state
- $gd_i(s) = \min_{a \in A(s)} (c(s,a) + \gamma \sum_{s' \in \text{succ}(s,a)} p(s'|s,a) gd_{i-1}(s'))$ if s is not a goal state
- $gd(s) = \lim_{i \rightarrow \infty} gd_i(s)$
- a(s) = the optimal action to execute in state s
- $a(s) = \text{argmin}_{a \in A(s)} (c(s,a) + \gamma \sum_{s' \in \text{succ}(s,a)} p(s'|s,a) gd(s'))$ if s is not a goal state
- gd(s) does not necessarily converge after a finite amount of time
- a(s) converges after a finite amount of time if gd(s) is approximated with $gd_i(s)$ for all s

example of Value Iteration (1)



i		0	1	2	3	4	5	6	7	
start	a1	0	3	1	3	1.9	3	2.5075	3	2.9631
	a2	1	1	1.9	2.5075	2.9631				
state 1	a3	0	1	1	1.45	1.45	1.855	1.855	2.1284	2.1284
		0	0	0	0	0	0	0	0	0
goal							3	3	3	3
							3.2912	3.4	3.4075	
							2.3334	2.3334	2.35	2.35
							0	0	0	0

example of Value Iteration (2)



which action to execute in the start state?

a1: $3 + 0 = 3$

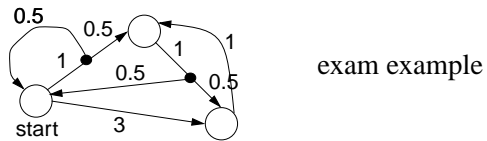
a2: $1 + 0.9 \cdot 0.5 \cdot 3 + 0.9 \cdot 0.5 \cdot 2.35 = 3.4075$

execute a1!

(In general, the optimal action depends on the discount factor!)

“learning” for optimization

“reinforcement learning” with Markov Decision Process Models



exam example

find a policy (behavior) that maximizes the expected total discounted reward even in the presence of delayed rewards

if you don't know the action outcomes (rewards and probabilities): reinforcement learning

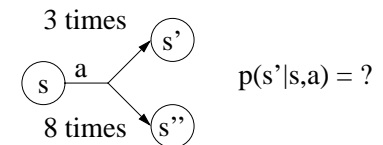
exploration/exploitation tradeoff

“learning” for optimization

“reinforcement learning” with Markov Decision Process Models

approach 1

estimate the probabilities and rewards



$p(s'|s,a) = ?$

use value-iteration

“learning” for optimization

“reinforcement learning” with Markov Decision Process Models

approach 2

use Q-learning

if you execute action a in state s and
you receive cost c and make a transition to state s'
then update

$$Q(s,a) = Q(s,a) + \alpha (c + \gamma V(s') - Q(s,a))$$

↑ learning rate
↑ discount factor
 $0 < \gamma < 1$

$$V(s') = \min_{a \in A(s')} Q(s',a)$$

$Q(s,a)$ = minimal expected discounted total cost until a goal is reached
if execution starts in state s and the first action executed is a
 $V(s')$ = minimal expected discounted total cost until a goal is reached
if execution starts in state s' (= “gd(s')”)

“learning” for optimization

“reinforcement learning” with Markov Decision Process Models

approach 2

1. Initialize $Q(s,a) = 0$ for all states s and actions a .
2. $s :=$ the current state.
3. if s is a goal state then stop.
4. Choose an action a to execute in the current state s .
(The action believed to be best is $a := \operatorname{argmin}_{a \in A(s)} Q(s,a)$.)
5. Execute action a . Observe the cost c and successor state s' .
6. Update $Q(s,a) = Q(s,a) + \alpha (c + \gamma V(s') - Q(s,a))$.
7. Goto 2.

