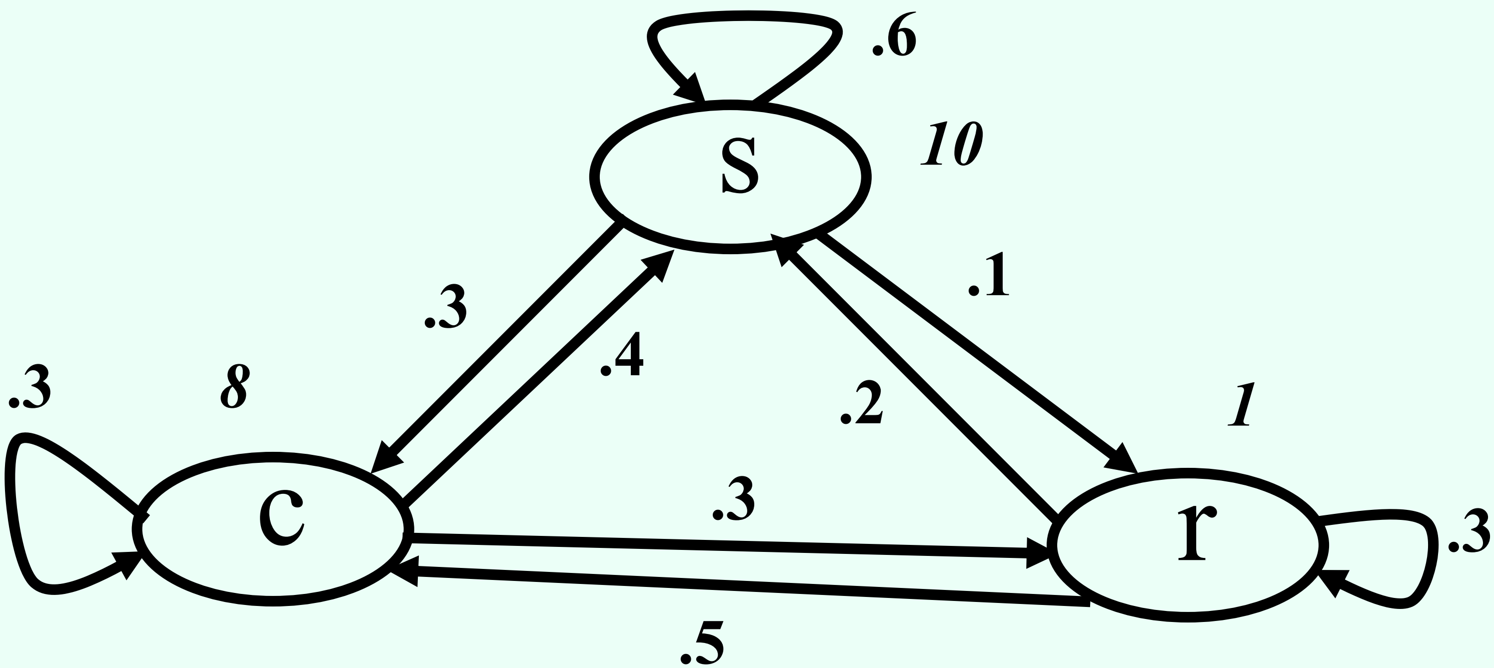
CPS 570: Artificial Intelligence

Markov decision processes, POMDPs

Instructor: Vincent Conitzer

Warmup: a Markov process with rewards

- We derive some reward R from the weather each day, but cannot influence it



- How much utility can we expect in the long run?
  - Depends on discount factor δ
  - Depends on initial state

A key equation

- Conditional expectation:

  $E(X \mid Y=y) = \Sigma_x\, x\, P(X=x|Y=y)$
- Let $P(s, s') = P(S_{t+1}=s' \mid S_t=s)$
- Let $v(s)$ be the (long-term) expected utility from being in state s now
- $v(s) = E(\Sigma_{t=0 \text{ to infinity}}\, \delta^t R(S_t) \mid S_0=s) =$

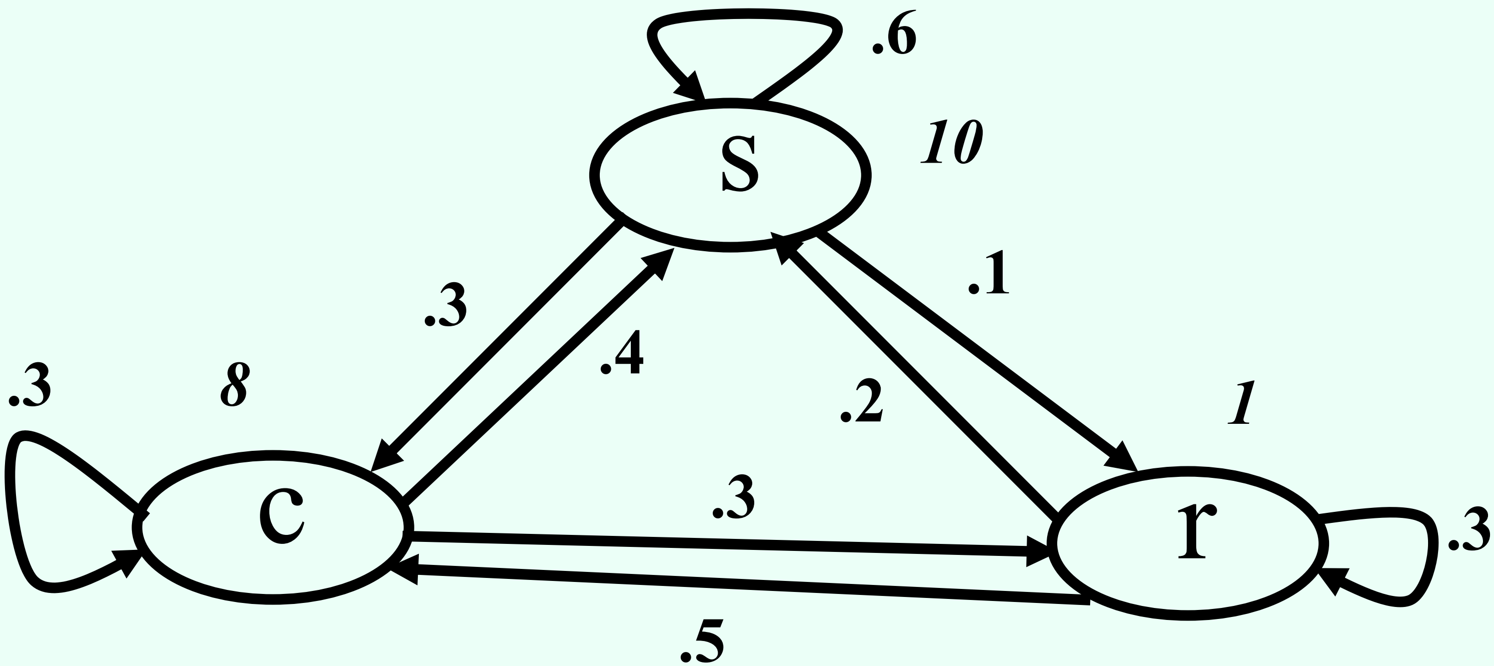
  $R(s) + \Sigma_{s'}\, P(s, s')\, E(\Sigma_{t=1 \text{ to infinity}}\, \delta^t R(S_t) \mid S_1=s')$
- But: $E(\Sigma_{t=1 \text{ to infinity}}\, \delta^t R(S_t) \mid S_1=s') =$

  $\delta E(\Sigma_{t=0 \text{ to infinity}}\, \delta^t R(S_t) \mid S_0=s') = \delta v(s')$
- We get: $v(s) = R(s) + \delta\Sigma_{s'}\, P(s, s')\, v(s')$

Figuring out long-term rewards

- Let v(s) be the (long-term) expected utility from being in state s now
- Let P(s, s') be the transition probability from s to s'
- We must have: for all s,

  $v(s) = R(s) + \delta \Sigma_{s'} P(s, s')\, v(s')$



- E.g., $v(c) = 8 + \delta(.4v(s) + .3v(c) + .3v(r))$
- Solve system of linear equations to obtain values for all states

Iteratively updating values

- If we do not want to solve system of equations…
  - E.g., too many states
- … can iteratively update values until convergence
- $v_i(s)$ is value estimate after i iterations
- $v_i(s) = R(s) + \delta\Sigma_{s'} P(s, s') v_{i-1}(s')$
- Will converge to right values
- If we initialize v_0=0 everywhere, then $v_i(s)$ is expected utility with only i steps left (finite horizon)
  - Dynamic program from the future to the present
  - Shows why we get convergence: due to discounting far future does not contribute much

Markov decision process (MDP)

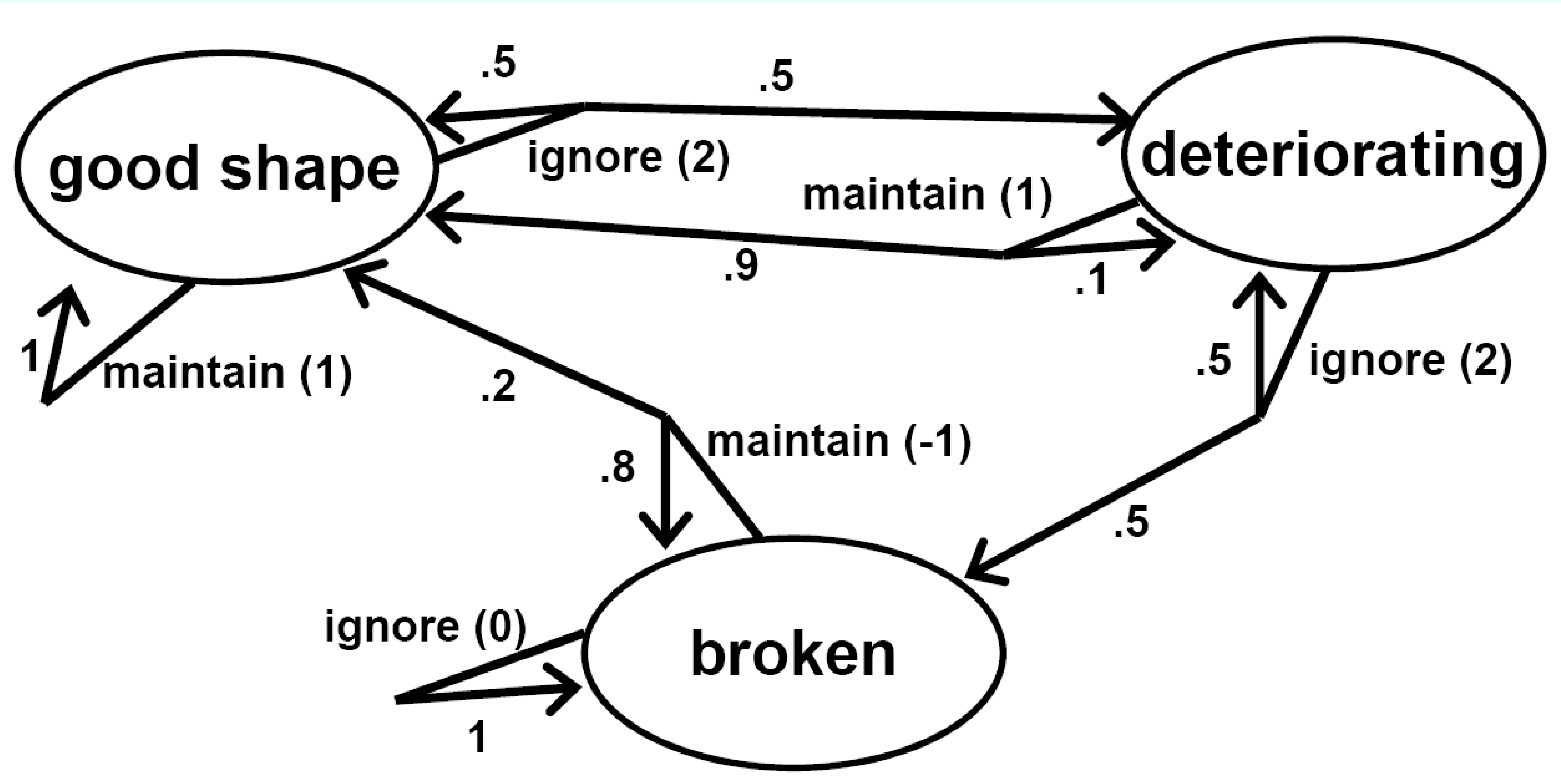
- Like a Markov process, except every round we make a decision
- Transition probabilities depend on actions taken

  $P(S_{t+1} = s' \mid S_t = s, A_t = a) = P(s, a, s')$
- Rewards for every state, action pair

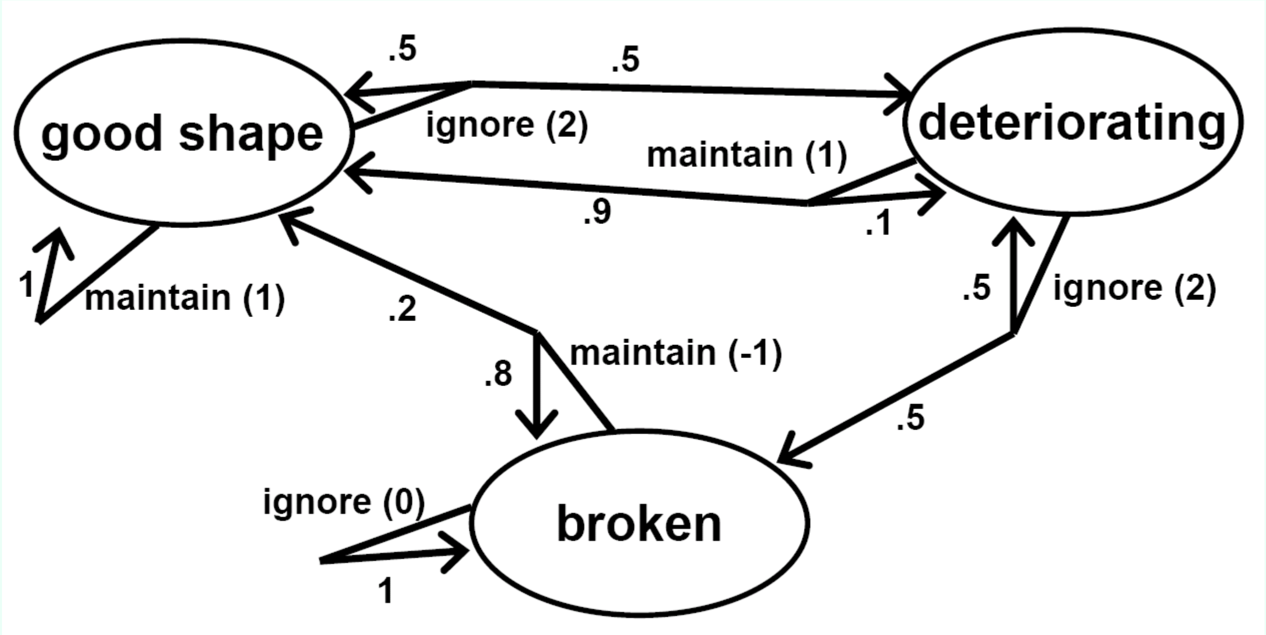
  $R(S_t = s, A_t = a) = R(s, a)$
  - Sometimes people just use $R(s)$; $R(s, a)$ little more convenient sometimes
- Discount factor δ

Example MDP

- Machine can be in one of three states: good, deteriorating, broken
- Can take two actions: maintain, ignore



Policies



- No time period is different from the others
- Optimal thing to do in state s should not depend on time period
  - … because of infinite horizon
  - With finite horizon, don't want to maintain machine in last period
- A policy is a function π from states to actions
- Example policy: π(good shape) = ignore, π(deteriorating) = ignore, π(broken) = maintain

Evaluating a policy

- Key observation: *MDP + policy = Markov process with rewards*
- Already know how to evaluate Markov process with rewards: system of linear equations
- Gives algorithm for finding optimal policy: try every possible policy, evaluate
  - Terribly inefficient

Bellman equation

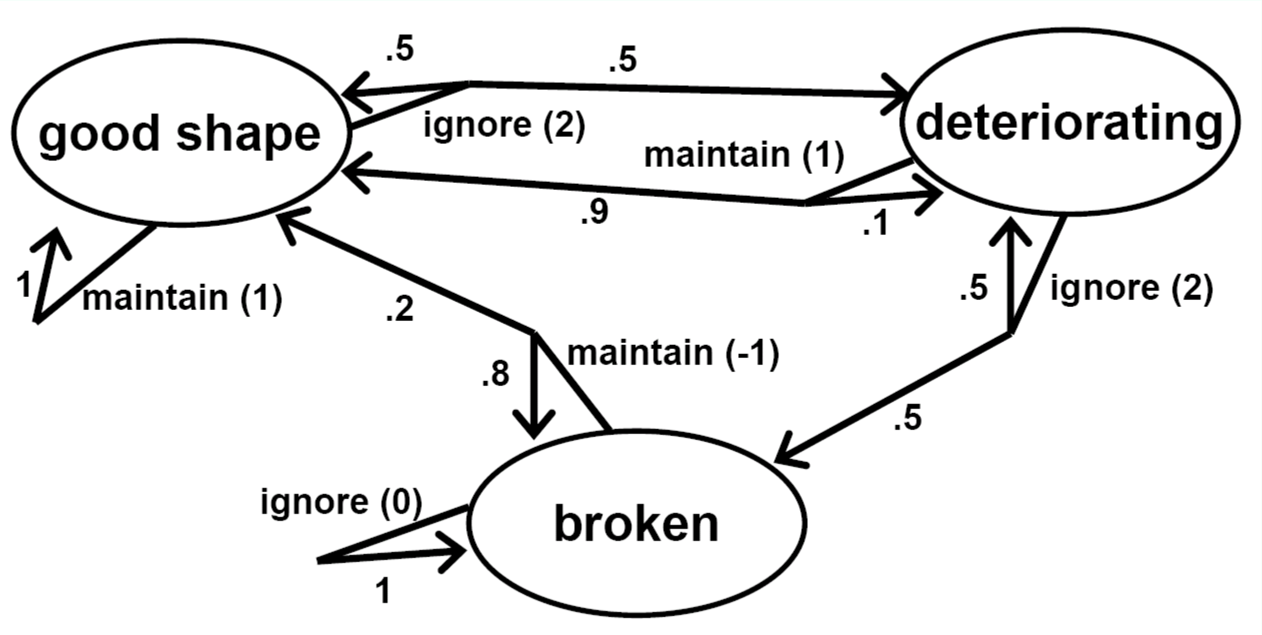
- Suppose you are in state s, and you play optimally from there on
- This leads to expected value $v^*(s)$
- Bellman equation:

  $v^*(s) = \max_a [R(s, a) + \delta\Sigma_{s'} P(s, a, s') v^*(s')]$
- Given v^*, finding optimal policy is easy

Value iteration algorithm for finding optimal policy

- Iteratively update values for states using Bellman equation
- $v_i(s)$ is our estimate of value of state s after i updates
- $v_{i+1}(s) = \max_a [R(s, a) + \delta \Sigma_{s'} P(s, a, s') v_i(s')]$
- Will converge
- If we initialize $v_0=0$ everywhere, then $v_i(s)$ is optimal expected utility with only i steps left (finite horizon)
  - Again, dynamic program from the future to the present

Value iteration example, δ=.9

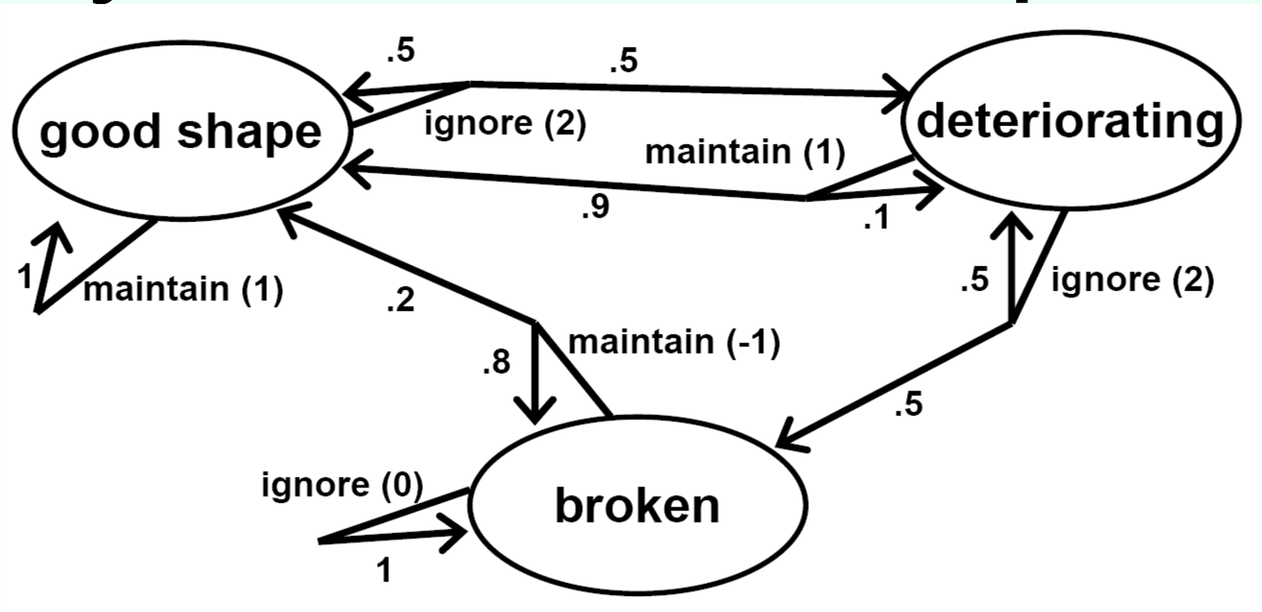


- $v_0(G) = v_0(D) = v_0(B) = 0$
- $v_1(G) = \max\{R(G,i) + \delta\Sigma_{s'} P(G, i, s') v_0(s'), R(G,m) + \delta\Sigma_{s'} P(G, m, s') v_0(s')\} = \max\{2,1\} = 2$;
- Similarly, $v_1(D)=\max\{2,1\} = 2$, $v_1(B) = \max\{0,-1\} = 0$
- $v_2(G) = \max\{R(G,i) + \delta\Sigma_{s'} P(G, i, s') v_1(s'), R(G,m) + \delta\Sigma_{s'} P(G, m, s') v_1(s')\} = \max\{2 + .9(.5v_1(G)+.5v_1(D)), 1 + .9(1v_1(G))\} = 3.8$;
- $v_2(D) = \max\{2 + .9(.5*2 + .5*0), 1 + .9(.9*2 + .1*2)\} = 2.9$
- $v_2(B) = \max\{0 + .9(1*0), -1 + .9(.8*0 + .2*2)\} = 0$
- Value for each state (and action at each state) will converge

Policy iteration algorithm for finding optimal policy

- Easy to compute values **given** a policy
  - No max operator
- Alternate between evaluating policy and updating policy:
- Solve for function v_i based on π_i
- $\pi_{i+1}(s) = \arg\max_a [R(s, a) + \delta \Sigma_{s'} P(s, a, s') v_i(s')]$
- Will converge

Policy iteration example, δ=.9



- Initial policy π_0: always maintain the machine
- Since we always maintain, the value equations become:

  $v_0(G) = 1+.9v_0(G)$; $v_0(D) = 1+.9(.9v_0(G)+.1v_0(D))$; $v_0(B) = -1+.9(.2v_0(G)+.8v_0(B))$
- Solving gives: $v_0(G) = 10$, $v_0(D)=10$, $v_0(B) = 2.9$
- Given these values, expected value for ignoring at G is 2 + .9(.5*10+.5*10)=11, expected value for maintaining at G is 1 + .9*10 = 10, so ignoring is better;
- For D, ignore gives 2 + .9(.5*10+.5*2.9) =7.8, maintain gives 1 + .9(.9*10+.1*10) = 10, so maintaining is better;
- For B, ignore gives 0 + .9*2.9, maintain gives -1 + .9(.2*10+.8*2.9)= 2.9, so maintaining is better;
- So, the new policy π_1 is to maintain the machine in the deteriorating and broken states only; solve for the values with π_1 , etc. until policy stops changing

Mixing things up

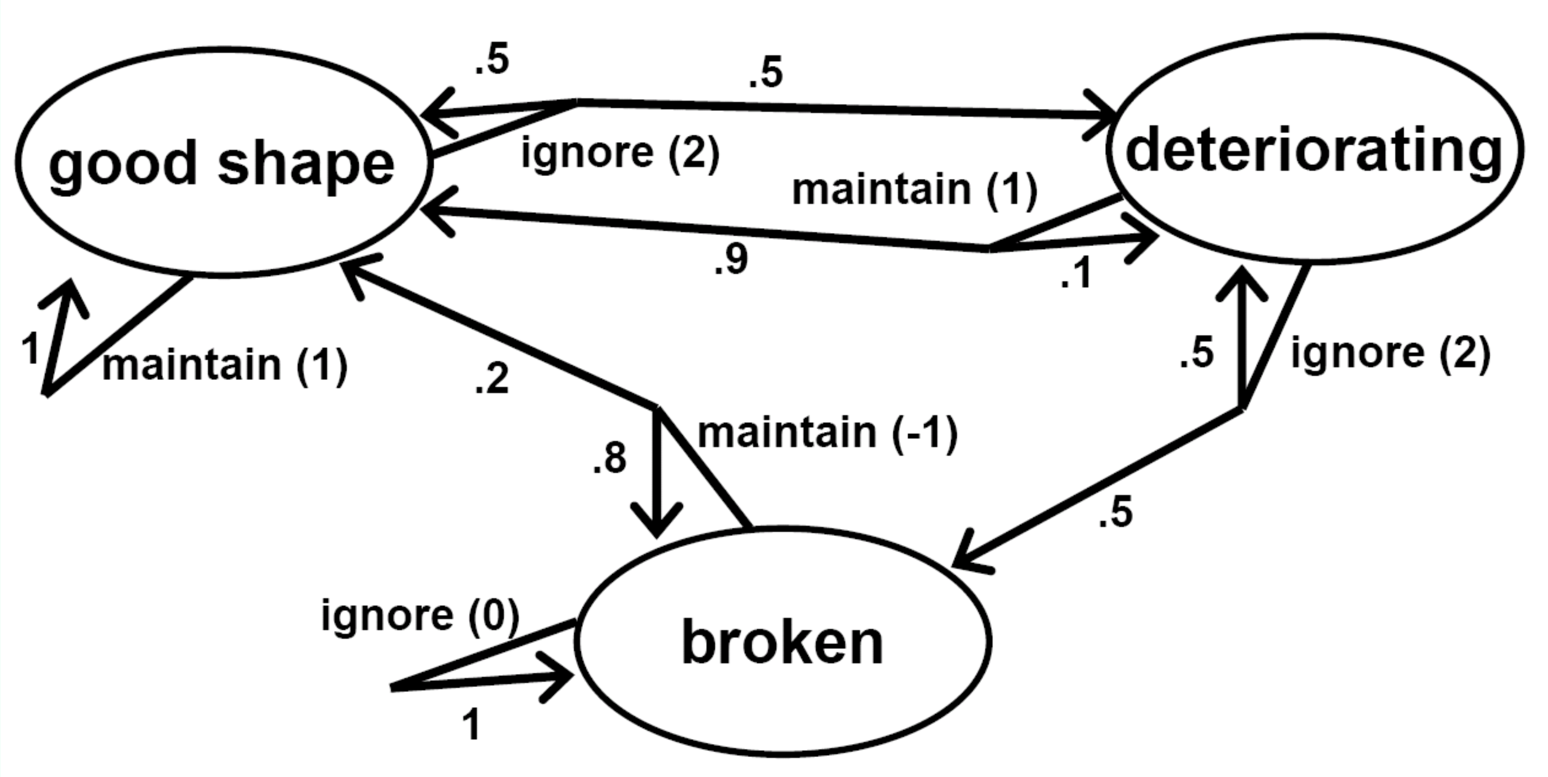
- Do not need to update every state every time
  - Makes sense to focus on states where we will spend most of our time
- In policy iteration, may not make sense to compute state values exactly
  - Will soon change policy anyway
  - Just use some value iteration updates (with fixed policy, as we did earlier)
- Being flexible leads to faster solutions

Partially observable Markov decision processes (POMDPs)

- Markov process + partial observability = HMM
- Markov process + actions = MDP
- Markov process + partial observability + actions = HMM + actions = MDP + partial observability = **POMDP**

	full observability	*partial observability*
no actions	**Markov process**	**HMM**
actions	**MDP**	**POMDP**

Example POMDP



- Need to specify observations
- E.g., does machine fail on a single job?
- P(fail | good shape) = .1, P(fail | deteriorating) = .2, P(fail | broken) = .9
  - Can also let probabilities depend on action taken

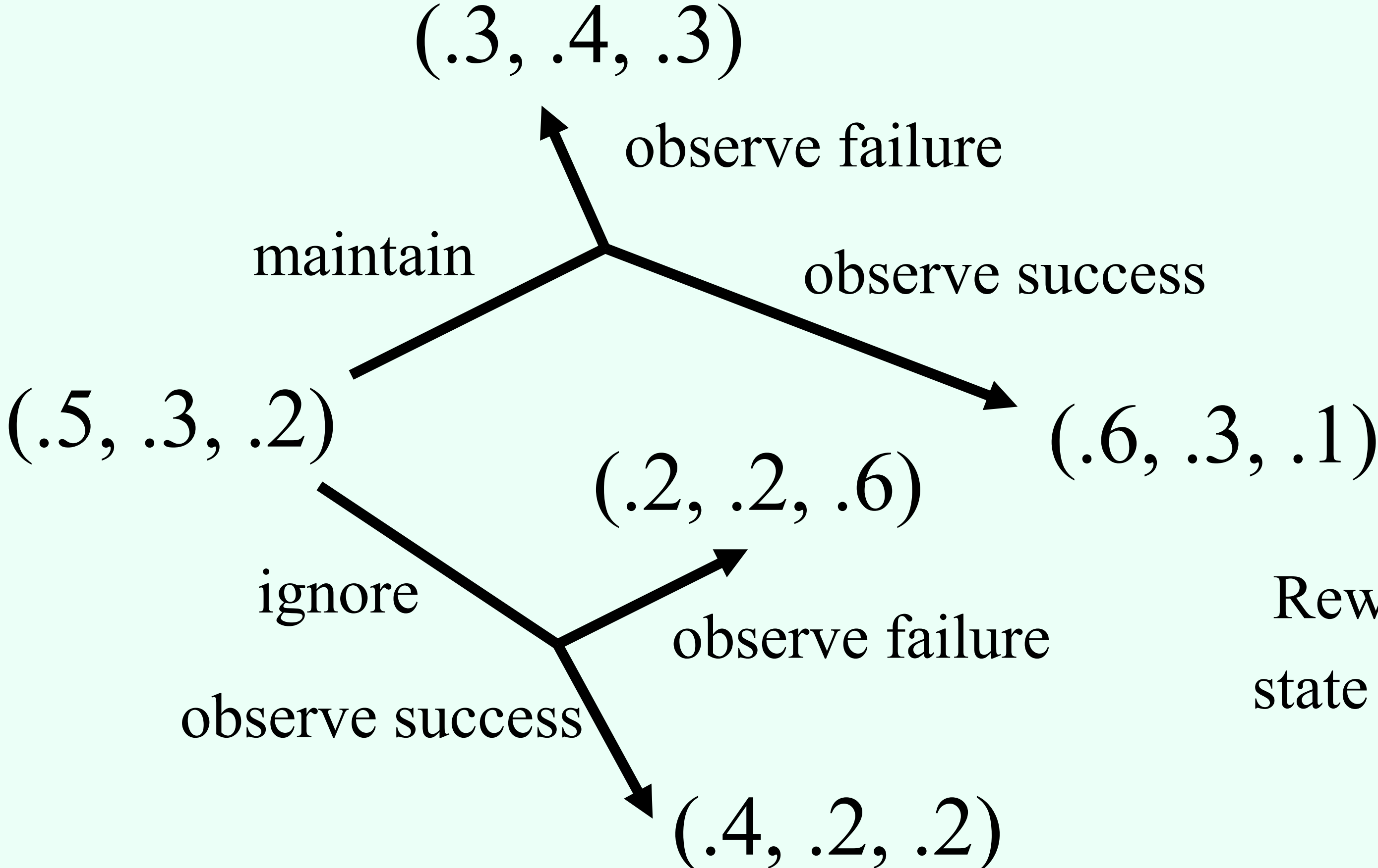
Optimal policies in POMDPs

- Cannot simply use $\pi(s)$ because we do not know s
- We can maintain a probability distribution over s:

  $P(S_t \mid A_1 = a_1, O_1 = o_1, \ldots, A_{t-1} = a_{t-1}, O_{t-1} = o_{t-1})$
- This gives a belief state b where b(s) is our current probability for s
- Key observation: *policy only needs to depend on* b, $\pi(b)$

Solving a POMDP as an MDP on belief states

- If we think of the belief state as the state, then the state is observable and we have an MDP



disclaimer: did not actually calculate these numbers…

Reward for an action from a state = expected reward given belief state

- Now have a large, continuous belief state…
- Much more difficult