Markov Decision Processes

Andrew Schaefer

EWO Seminar

October 26, 2006

What are Markov Decision Processes (MDPs)?

- MDPs are a method for formulating and solving stochastic and dynamic decisions
- MDPs are very flexible, which is an advantage from a modeling perspective but a drawback from a solution viewpoint (can’t take advantage of special structure)
- MDPs are pervasive; every Fortune 500 company uses them in some form or another
- Particular success in inventory management and reliability

Potential for MDPs in EWO

- The flexibility of MDPs may allow them to be useful for certain problems
- It is unlikely that they can work as stand-alone techniques
- Long history of successful applications

Outline

- Basic Components of MDPs
- Solution Techniques
- Extensions
- Big Picture – Potential Roles for MDPs

Outline – Basic Components

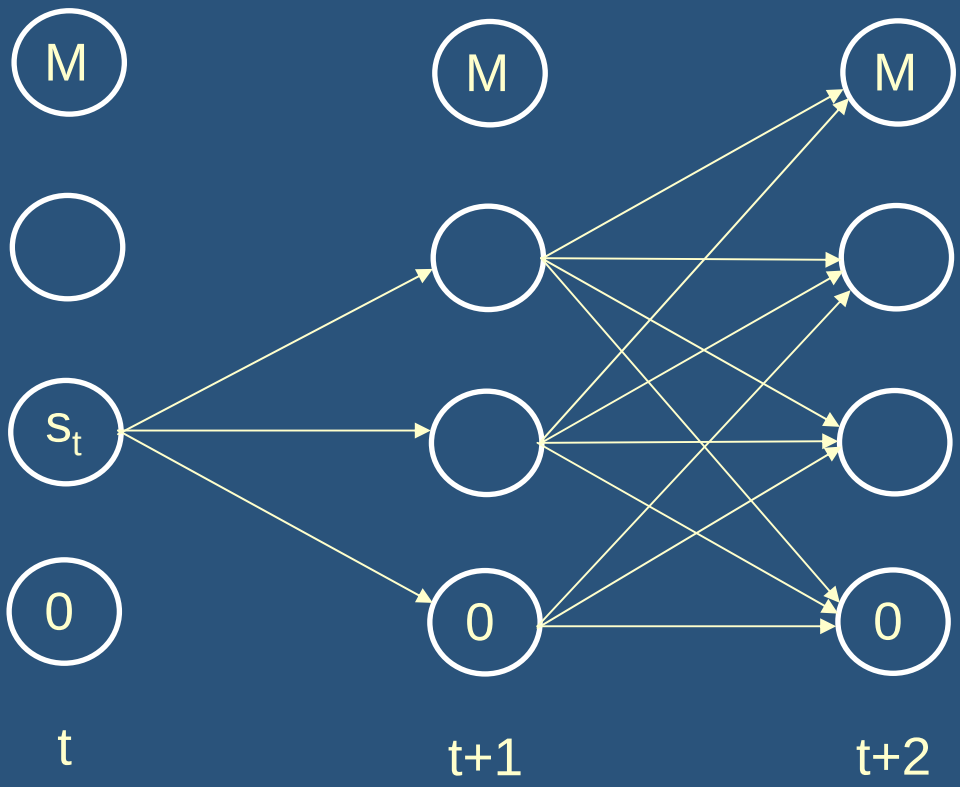
- Inventory Example

- MDP Vocabulary
  - 5 Basic Components
    1. decision epochs
    2. states
    3. actions
    4. rewards
    5. transition probabilities
  - value function
  - decision rule
  - policy

Inventory Example

- Problem Statement
  - Each month, the manager of a warehouse
    - observes current inventory on hand of a single product
    - decides how much additional stock to order from a supplier
  - Monthly demand
    - uncertain
    - probability distribution is known
  - Tradeoff between costs of
    - keeping inventory
    - lost sales
  - Objective
    - maximize expected profit over the next year
- Simplifying Assumptions
  - Instantaneous delivery
  - No backlogging, i.e. excess demand is lost
  - Warehouse capacity of M units
  - Product is sold in whole units only
  - Revenues, costs and demand distribution do not change from month to month

Inventory Level



$$s_{t+1} = s_t + a_t - \min\{D_t, s_t + a_t\}$$

Outline – Basic Components

- Inventory Example
- MDP Vocabulary
  - 5 Basic Components
    1. decision epochs
    2. states
    3. actions
    4. rewards
    5. transition probabilities
  - value function
  - decision rule
  - policy

1. Decision Epochs: t

- **Points in time at which decisions are made**
  - analogous to period start times in a “Markov Process”
- Inventory example
  - first day of month 1, first day of month 2, …, first day of month 12
- In general: $1, 2, \ldots, N$
  - N: length of the time horizon (could be infinite)
- Also called
  - periods
  - stages

2. States: s_t

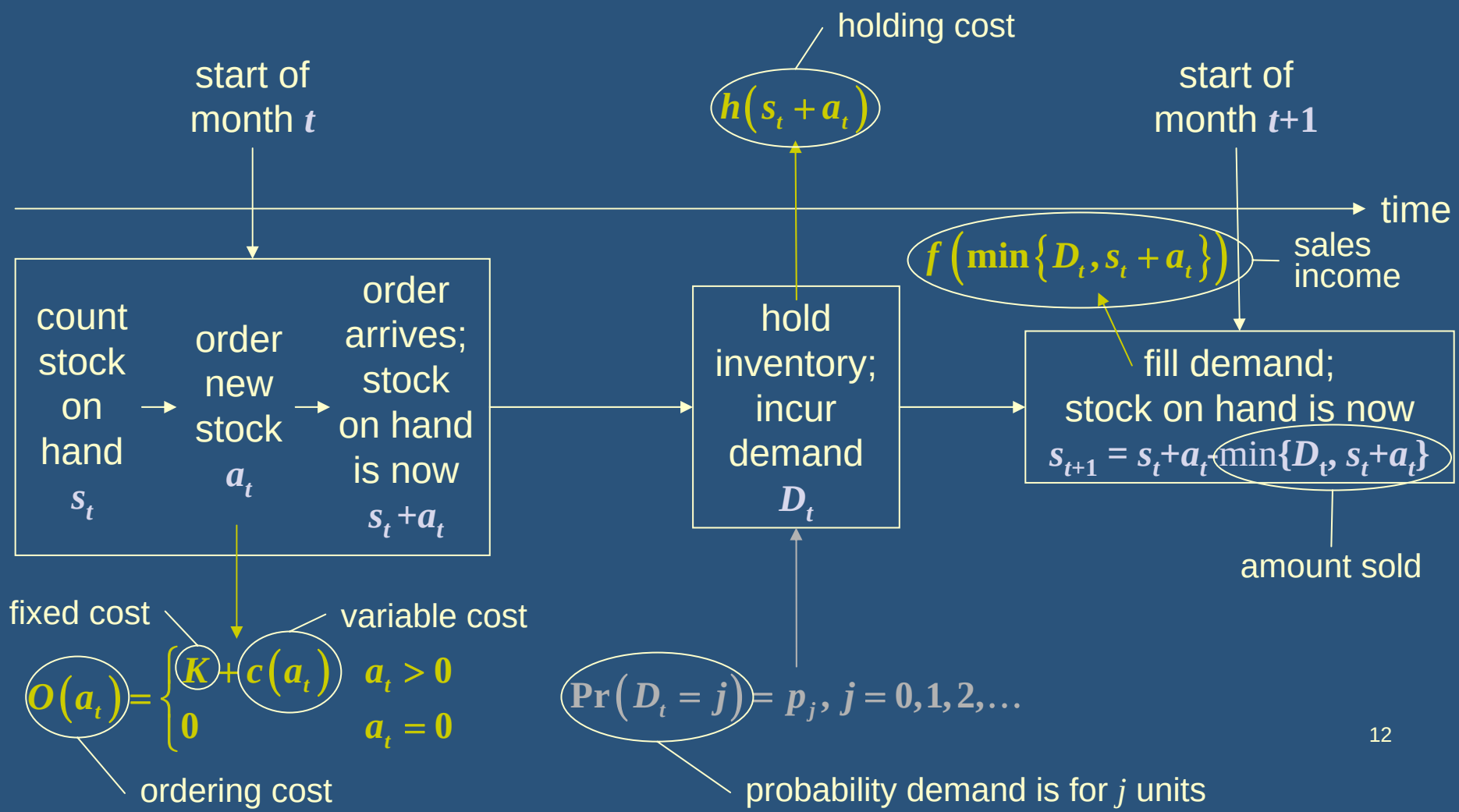
- **Relevant information needed to describe the system**
  - analogous to states in “Markov Processes”
  - includes all information from the past relevant to the future

- Inventory example
  - amount of inventory on hand at the start of the month
    - s_t, $t = 1, \ldots, 12$
  - possible values – state space
    - $0, 1, 2, \ldots, M$

3. Actions: a_t

- **Means by which the decision maker interacts with the system**
  - permissible actions can be state dependent
  - no exact analogy to “Markov Processes”
    - the decision is usually modeled ***outside*** the Markov Process
- Inventory example
  - how much additional stock to order each month
  - denoted
    - a_t, $t = 1, \ldots, 12$
  - possible values
    - $0, 1, 2, \ldots, M - s_t$

Inventory Example



4. Rewards: $r_t(s_t,a_t)$

- **Expected immediate net income associated with taking a particular action, in a particular state, in a particular epoch**
  - analogous to state utilities in “Markov Processes”

- Inventory example
  - expected income – order cost – holding cost
  - $r_t(s_t,a_t) = \mathbf{E}[\text{income in month } t] - O(a_t) - h(s_t + a_t),\ t = 1,2,\ldots,12$

$$\mathbf{E}[\text{income in } t] = \mathbf{E}[\text{income in } t \mid D_t \le s_t + a_t]\Pr\{D_t \le s_t + a_t\} + \mathbf{E}[\text{income in } t \mid D_t > s_t + a_t]\Pr\{D_t > s_t + a_t\}$$

$$= \sum_{j=0}^{s_t+a_t} f(j)\,p_j + f(s_t + a_t) \sum_{j=s_t+a_t+1}^{\infty} p_j$$

  - in the last epoch, we assume that whatever is left over has some salvage value, $g(.)$

$r_{13}(s_{13},\cdot) = g(s_{13})$ in general $r_{N+1}(s_{N+1},\cdot) = g(s_{N+1})$

5. Transition Probabilities: $p_t(s_t,a_t)$

- **Distribution that governs how the state of the process changes as actions are taken over time**
  - depends on current state and action only, and possibly time
  - that is, the future is independent of the past given the present (The Markov Property)

- Inventory example
  - we already established that $s_{t+1} = s_t + a_t - \min\{D_t, s_t + a_t\}$

$$\underbrace{\Pr\{s_{t+1} = j \mid s_t = s, a_t = a\}}_{\text{depends on demand}} = \begin{cases} p_{s+a-j} & j \le s+a \\ \sum_{i=s+a}^{\infty} p_i & j = 0 \\ 0 & j > s+a \end{cases}$$

- $j \le s+a$: end up with some leftovers if demand is less than inventory
- $j = 0$: end up with nothing if demand exceeds inventory
- $j > s+a$: can't end up with more than you started with

Value Function

- Collectively, decision epochs, states, actions, rewards, and transition probabilities form an MDP.

- But how do they fit together?

- Value function, $v_t(s_t)$
  - **maximum total expected reward starting in state s_t with N-t decision epochs remaining**
  - $v_t(s_t) = \max_a\{r_t(s_t,a_t) + E\, v_{t+1}(s_{t+1})\}$
    - $r_t(s_t,a_t)$: expected immediate reward in period t
    - $E\, v_{t+1}(s_{t+1})$: expected remaining reward in periods $t + 1, t + 2,\ldots N$

Value Function

- Inventory example

$$v_{13}(s_{13}) = r_{13}(s_{13},\cdot) = g(s_{13})$$

$$v_{12}(s_{12}) = \max_{a_{12}\in\{0,1,\ldots,M-s_{12}\}}\left\{r_{12}(s_{12},a_{12}) + \sum_{j=0}^{M} v_{13}(j)\Pr\{s_{13}=j \mid s_{12},a_{12}\}\right\}$$

$$v_{11}(s_{11}) = \max_{a_{11}\in\{0,1,\ldots,M-s_{11}\}}\left\{r_{11}(s_{11},a_{11}) + \sum_{j=0}^{M} v_{12}(j)\Pr\{s_{12}=j \mid s_{11},a_{11}\}\right\}$$

$$\vdots$$

$$v_2(s_2) = \max_{a_2\in\{0,1,\ldots,M-s_2\}}\left\{r_2(s_2,a_2) + \sum_{j=0}^{M} v_3(j)\Pr\{s_3=j \mid s_2,a_2\}\right\}$$

$$\rightarrow\ v_1(s_1) = \max_{a_1\in\{0,1,\ldots,M-s_1\}}\left\{r_1(s_1,a_1) + \sum_{j=0}^{M} v_2(j)\Pr\{s_2=j \mid s_1,a_1\}\right\}$$

maximum total expected reward starting in state s_1 with 12 decision epochs remaining

expected immediate reward period 1

expected remaining reward in periods $(2,\ldots,N)$

Decision Rule

- **A rule, for a particular state, that prescribes an action for each decision epoch**
  - $d_t(s)$ = action to take in state s in decision epoch t

- Inventory example - possible decision rules
  - $d_1(0) = 5$:
    if there are 0 items on hand at the beginning of month 1, order 5.
  - $d_2(1) = 3$:
    if there is 1 item on hand at the beginning of month 2, order 3.

Policy

- **A collection of decision rules for all states**

- Inventory example

$$\begin{bmatrix} d_1(0) & d_2(0) & \dots & d_{12}(0) \\ d_1(1) & d_2(1) & \dots & d_{12}(1) \\ \vdots & \vdots & \ddots & \vdots \\ d_1(M) & d_2(M) & \dots & d_{12}(M) \end{bmatrix} = \begin{bmatrix} 5 & 4 & \dots\dots\dots & 1 \\ 4 & 3 & \dots\dots\dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots\dots\dots & 0 \end{bmatrix}$$

- Under a fixed policy, the process behaves according to a Markov chain.

Outline – Solution Techniques

- Finite-horizon MDPs
  - Backwards induction solution technique
- Infinite-horizon MDPs
  - Optimality criteria
  - Solution techniques
    - value iteration
    - policy iteration
    - linear programming

Various factors to consider in choosing the right technique

- Will rewards be discounted?
- What is the objective?
  - Maximize expected *total* reward, or
  - Maximize expected *average* reward
- Is the problem formulated as a finite or infinite horizon problem?
  - If finite horizon, solution technique does not depend on discounting or objective
  - For infinite horizon, technique does depend on discounting and objective

Outline – Solution Techniques

- Finite-horizon MDPs
  - Backwards induction solution technique
- Infinite-horizon MDPs
  - Optimality criteria
  - Solution techniques
    - value iteration
    - policy iteration
    - linear programming

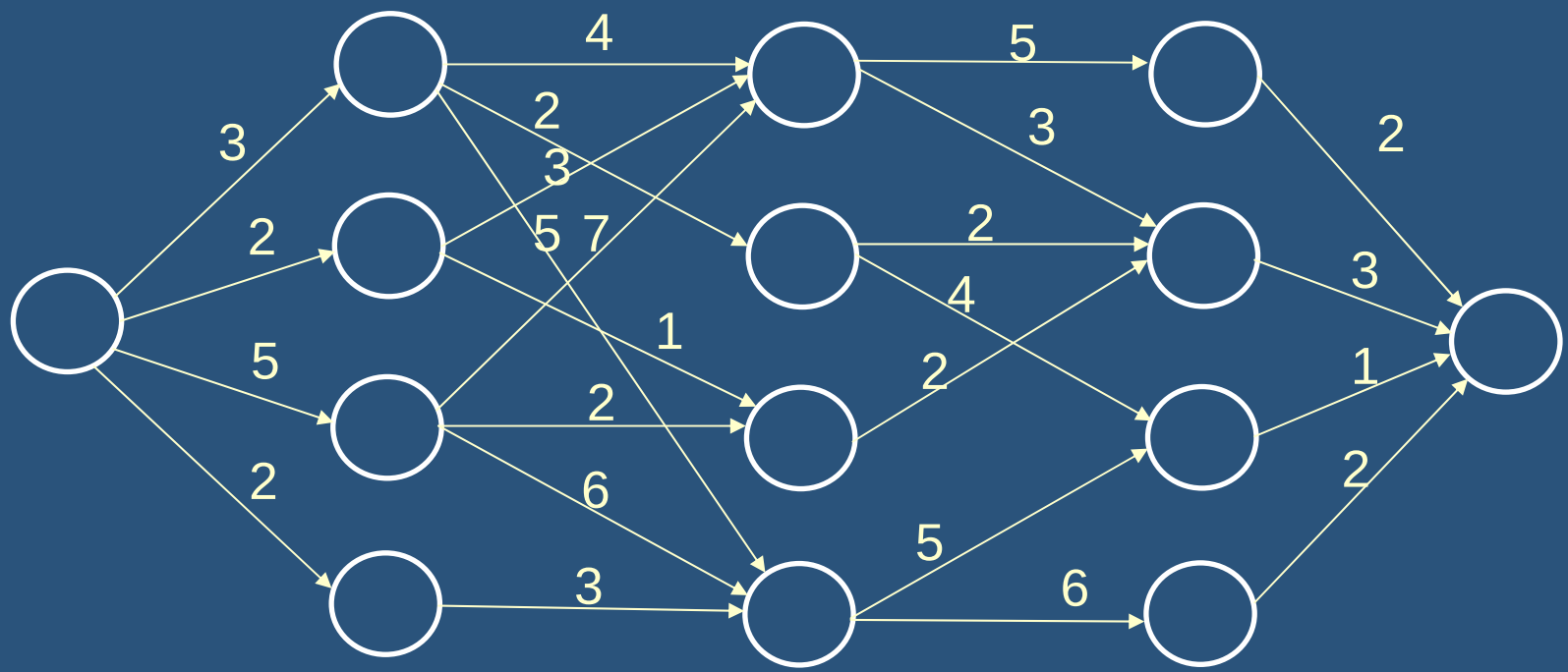
Finite horizon problems

- We want to find a policy, π , that maximizes:

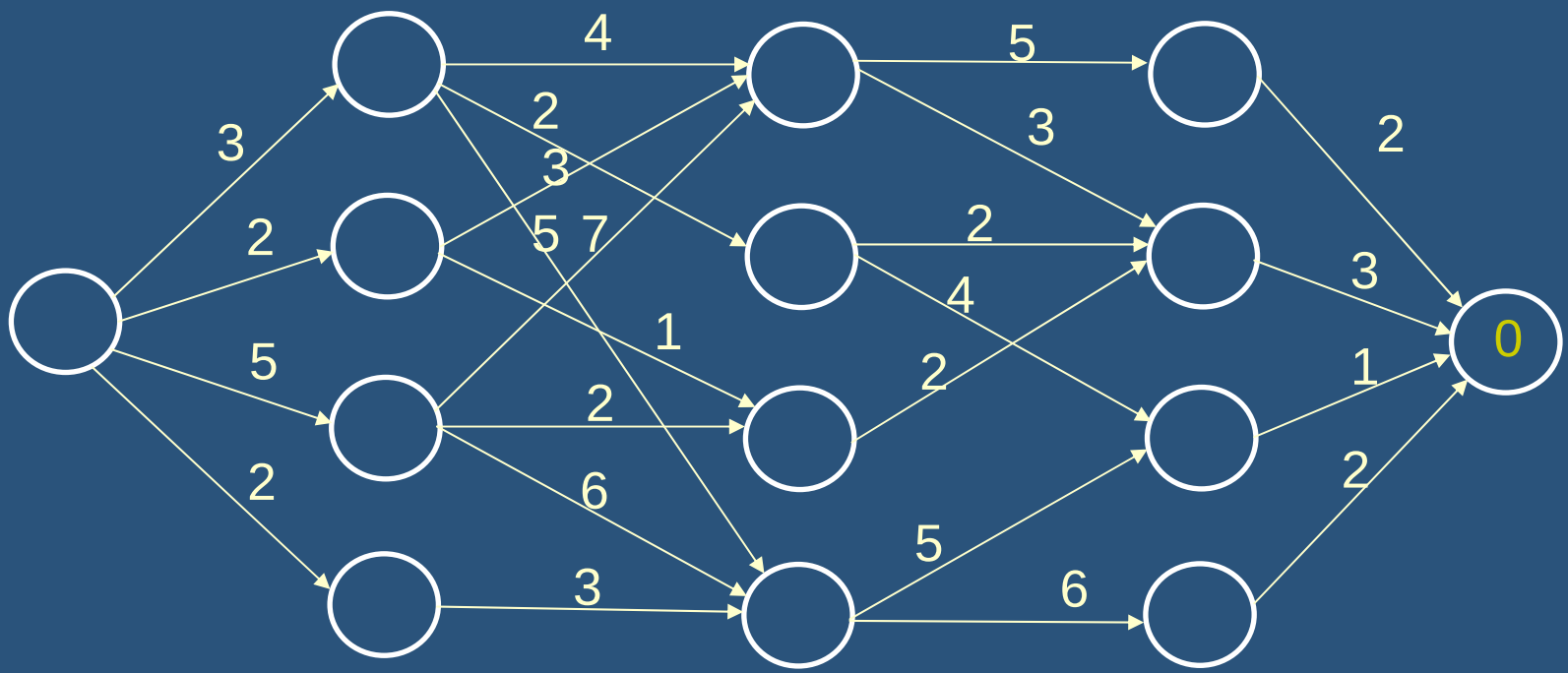
$$v_\pi(s) = \mathrm{E}_{\pi,s}\left\{\sum_{t=1}^{N} r(s_t, a_t)\right\}$$

- The solution technique used is called “backwards induction”

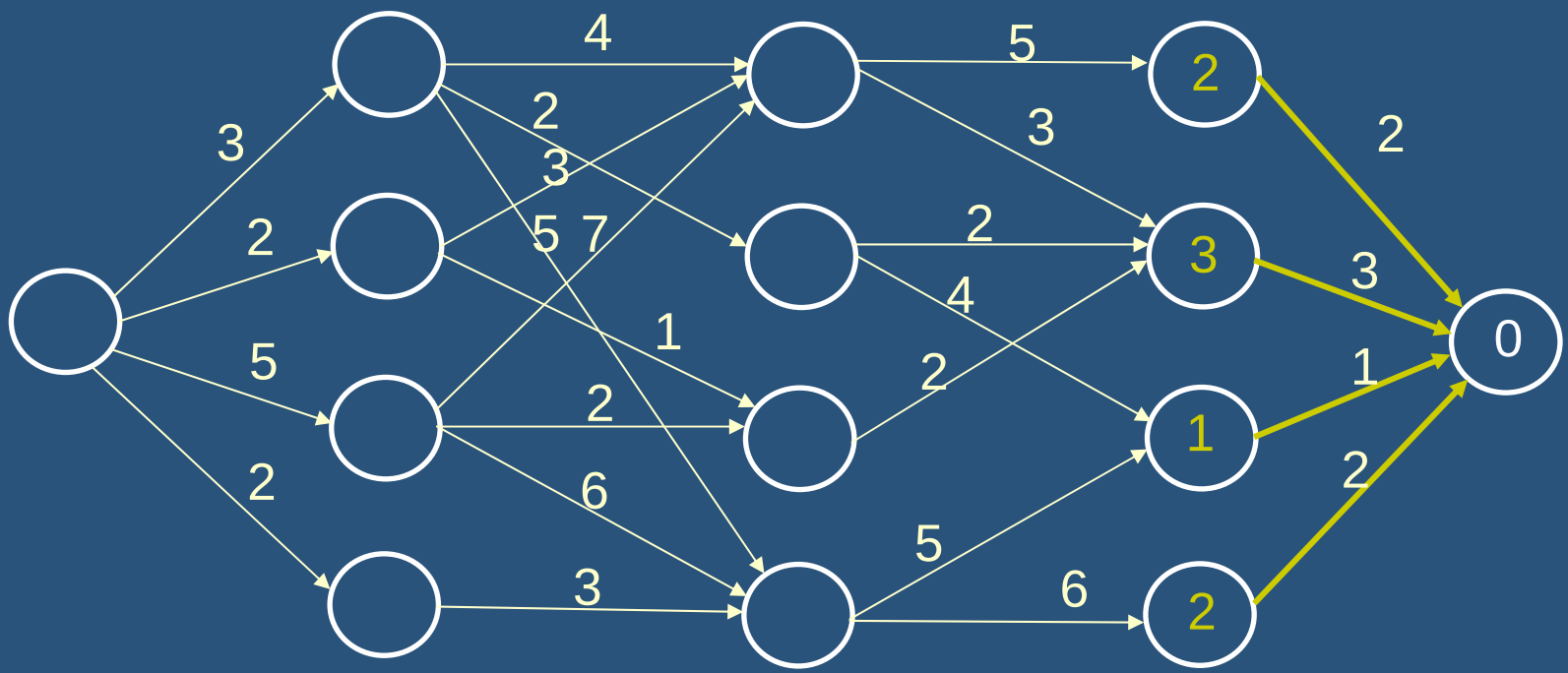
Backwards Induction – Shortest Path Problem



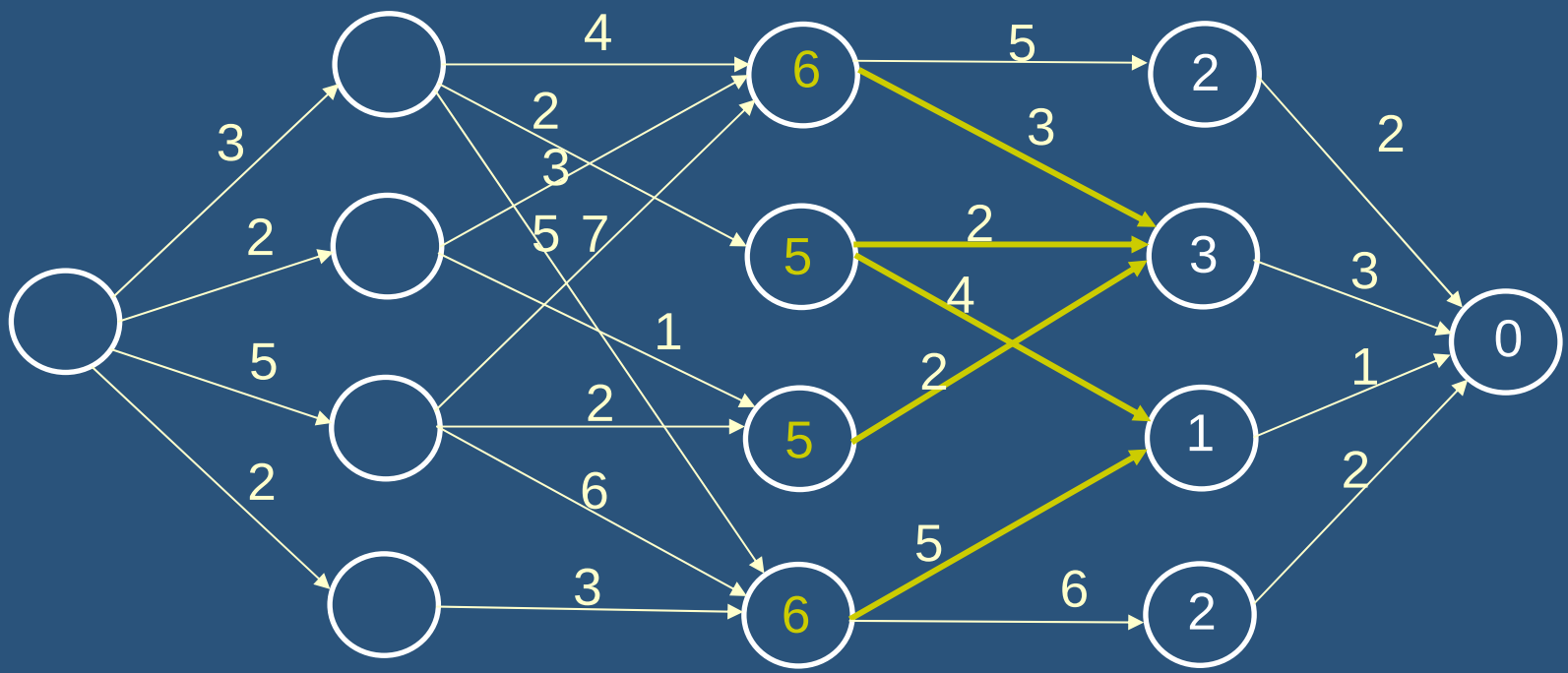
Backwards Induction – Shortest Path Problem



Backwards Induction – Shortest Path Problem



Backwards Induction – Shortest Path Problem



Idea behind backwards induction

- Envision being in the last time period for all the possible states and decide the best action for those states
- This yields an optimal value for that state in that period
- Next envision being in the next-to-last period for all the possible states and decide the best action for those states, *given you now know the optimal values of being in various states at the next time period*
- Continue this process until you reach the present time period

Formalization of Backwards Induction

- Assume N period horizon with the following parameters:
  - r(s,a): immediate reward for choosing action a when in state s
  - p(j|s,a): probability that system moves to state j at next time period given the current state is s and action a is chosen
  - g(s): salvage value of system occupying state s at final period N
- Algorithm:

1. Let $v_N(s) = g(s)$ for all s

2. For t = N-1, N-2, …, 1 and for all s at each period do:

a. $$v_t(s) = \max_{a \in A} \left\{ r(s,a) + \sum_{j \in S} p(j \mid s,a) v_{t+1}(j) \right\}$$

b. $$a_t(s) = \arg\max_{a \in A} \left\{ r(s,a) + \sum_{j \in S} p(j \mid s,a) v_{t+1}(j) \right\}$$

Infinite Horizon Problems

- The infinite horizon case is the limiting value of the finite case as the time horizon tends toward infinity
  - Recall the finite horizon value function:

$$v_\pi(s) = \mathrm{E}_{\pi,s}\left\{\sum_{t=1}^{N} r(s_t, a_t)\right\}$$

  - This is the infinite horizon value function:

$$v_\pi(s) = \lim_{N\to\infty} \mathrm{E}_{\pi,s}\left\{\sum_{t=1}^{N} r(s_t, a_t)\right\}$$

  - This is the infinite horizon value function with discounting:

$$v_\pi(s) = \lim_{N\to\infty} \mathrm{E}_{\pi,s}\left\{\sum_{t=1}^{N} \lambda^{t-1} r(s_t, a_t)\right\}$$

Important Factors for Infinite Horizon Models

- Discount factor
  - If we have a discount factor < 1, then total expected value converges to a finite solution
  - If there is no discounting, then total expected value may explode
    - However, if the system has an absorbing state with 0 reward, then even these may yield finite optimal values
- Objective criteria
  - Total expected value
  - Average expected value over the long run
    - More detailed analyses is required
- When is a stationary optimal policy guaranteed to exist?
  - If the state space and action space are finite

Previous applications of MDPs

- At the heart of every inventory management paper is an MDP
- Reliability and replacement problems
- Some routing and logistics problems (e.g. Warren Powell's work in trucking)
- 50 years of successful applications (particularly inventory theory, which is widely applied)

Limitations of MDPs

- "Curse of dimensionality"
  - As the problem size increases, i.e. the state and/or action space become larger, it becomes computationally very difficult to solve the MDPs
  - There are some methods that are more memory-efficient than Policy Iteration and Value Iteration algorithms
  - There are also some solution techniques that find near-optimal solutions in a short time
- Enormous data requirements
  - For each action and state pair, we need a transition probability matrix and a reward function

Limitations of MDPs

- Stationarity assumption
  - In fact, transition probabilities may not stay the same over time
  - Two methods to model non-stationary transition probabilities and rewards
    1. Use a finite-horizon model
    2. Enlarge the state space by including time

Why do MDPs not Scale Well?

- Analogy in the deterministic world: DP vs. IP
- Any integer program can be formulated as a dynamic program
- However, DPs struggle with IPs that take branch-and-bound less than a second
- Why?
  - IP uses polyhedral theory to get strong bounds
  - DP techniques grow with the size of the problem
- Will this occur in the stochastic setting?

MDP: Approximate Solution Techniques

- Finding the optimal policy is polynomial time in the state space size
  - Linear programming formulation
  - Sometimes too many states for this to be practical

- Examples of approximate solution techniques
  - State aggregation and action elimination
  - Many AI techniques (reinforcement learning)
  - Approximate linear programming (de Farias and van Roy)
  - In the last 10 years this has become a very active area

Extensions of MDPs

- Thus far, we've discussed discrete-time MDP, i.e. the decisions are made at certain time periods

- Continuous time MDP models generalize MDPs by
  - allowing the decision maker to choose actions whenever the system changes
  - modeling the system evolution in continuous time
  - allowing the time spent in a particular state to follow an arbitrary probability distribution
- No one has considered a continuous time stochastic program

Extensions of MDPs

- Most models are completely observable MDPs, i.e. the state of the system is known with certainty at all time periods

- Partially observable MDP models generalize MDPs by relaxing the assumption that the state of the system is completely observable

Partially Observable MDPs (POMDPs)

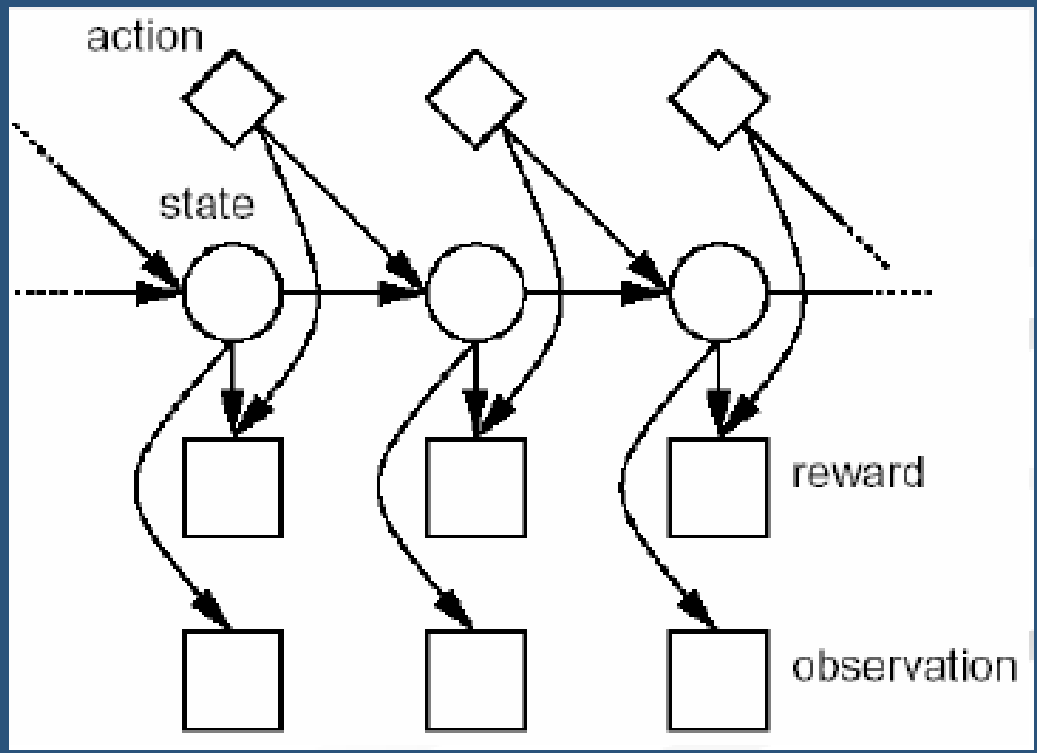
- System state is not fully observable.
  - Decision maker receives a signal o which is related to the system state by *q(o|s)*
- Observation process, depends on the (unknown) state, generated from a probability distribution function
- Hidden variables form a standard MDP
- Applications
  - Medical diagnosis and treatment
  - Equipment repair

POMDPs- Formal Definition

- POMDPs consist of
  - State space S
  - Action space A
  - Observation space O
  - Transition matrix P
  - Rewards R
  - Observation process distribution $Q(o \mid s)$
  - Value function $V = V(y; \pi)$
    - Where $\pi: O \rightarrow A$ is a policy

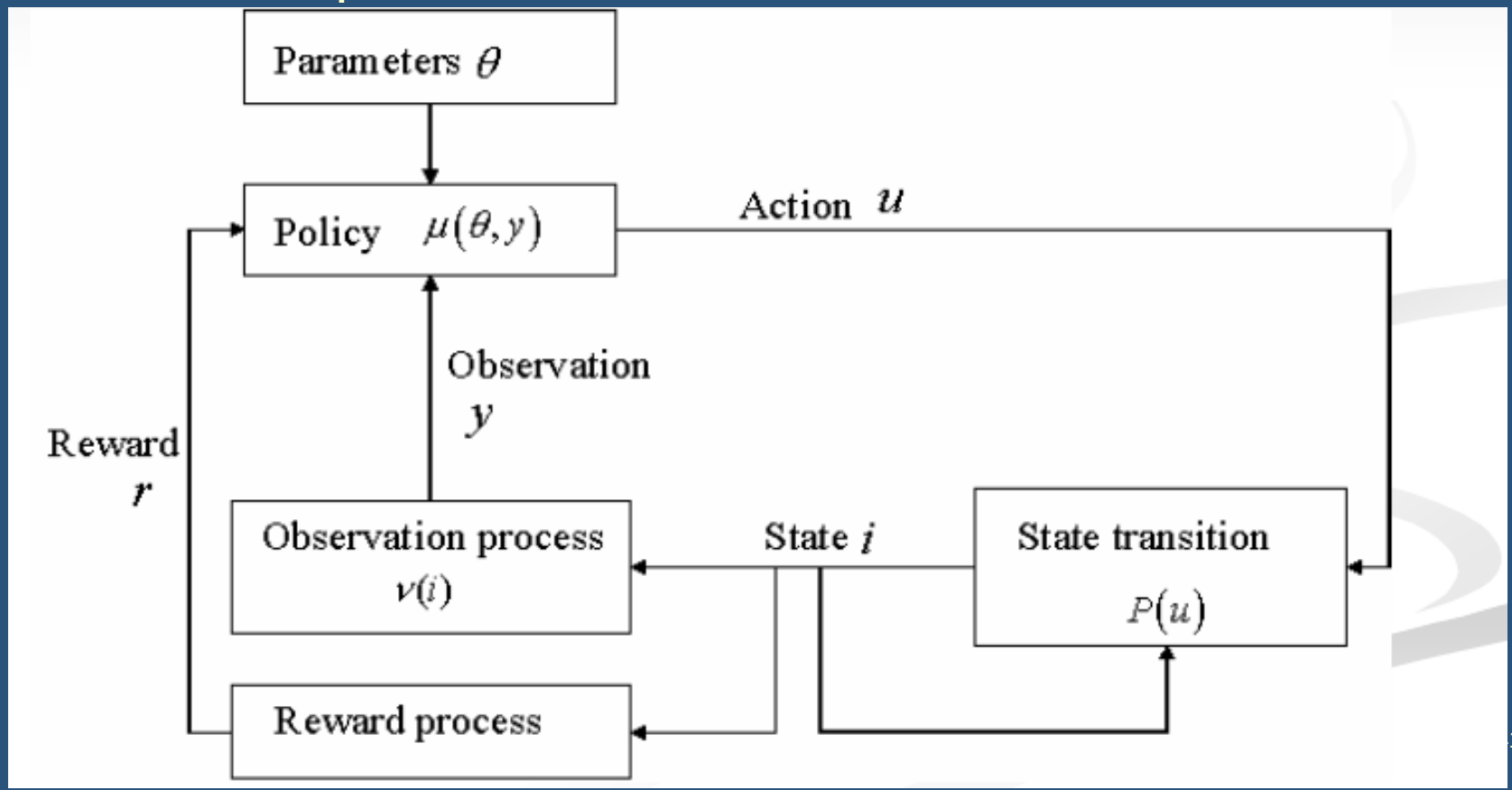
POMDP - Contd

- Simple example



POMDP controlled by parameterized stochastic policy

- Dynamics of POMDP controlled by parameterized stochastic policies:



Constrained Markov Decision Process

- MDPs with probabilistic constraints
  - Typically linear in terms of the limiting probability variables
- Applications
  - Service level constraints in call centers
    - P(No delay) >= 0.5
    - E[Number in queue] <= 10
- Value iteration and Policy iteration algorithms do not extend. Linear Programming does.

CMDP: LP technique

- Can model constraint as a function of the variables
- Results in the optimal randomized policy
  - Optimal choice of action in some states can be randomized
  - Not practical
  - Difficult to obtain the actual randomization probabilities
- Why randomized?
  - Vertices of LP polytope without constraints correspond to non-randomized policies
  - Constraints create new vertices corresponding to randomized policies

Competitive MDPs

- Two (or more) players compete
- The actions of one player affects the state of the other
- May be interesting for modeling various oligopolistic relationships among competitors
- Still a nascent area

Why would I prefer an MDP to a Stochastic Program?

- MDPs are much more flexible – existing multistage SP models assume linear relations across stages
- If there is a recursive nature to the problem, MDPs are likely superior
- When the number of states/actions is small relative to the time horizon (e.g. inventory management) MDPs are likely to be superior
- The MDP may be easier to analyze analytically; e.g. the optimality of (s,S) policies in inventory theory

Why would I prefer a Stochastic Program to an MDP?

- When the action and/or state space has a (good) polyhedral representation
- When the set of decisions and/or states is enormous, or infinite
- When the number of stages is not critical relative to the detail required in each stage

Conclusions

- MDPs are a flexible technique for stochastic and dynamic optimization problems
- MDPs have a much longer history of success than stochastic programming
- Curse of dimensionality may render them impractical (e.g. for a VRP, action space is set of TSP tours)
- Approximate solution techniques appear promising
- Allow models that are unexplored in SP
  - Partial observability
  - Continuous time
  - Competition