Advanced Predictive Modeling

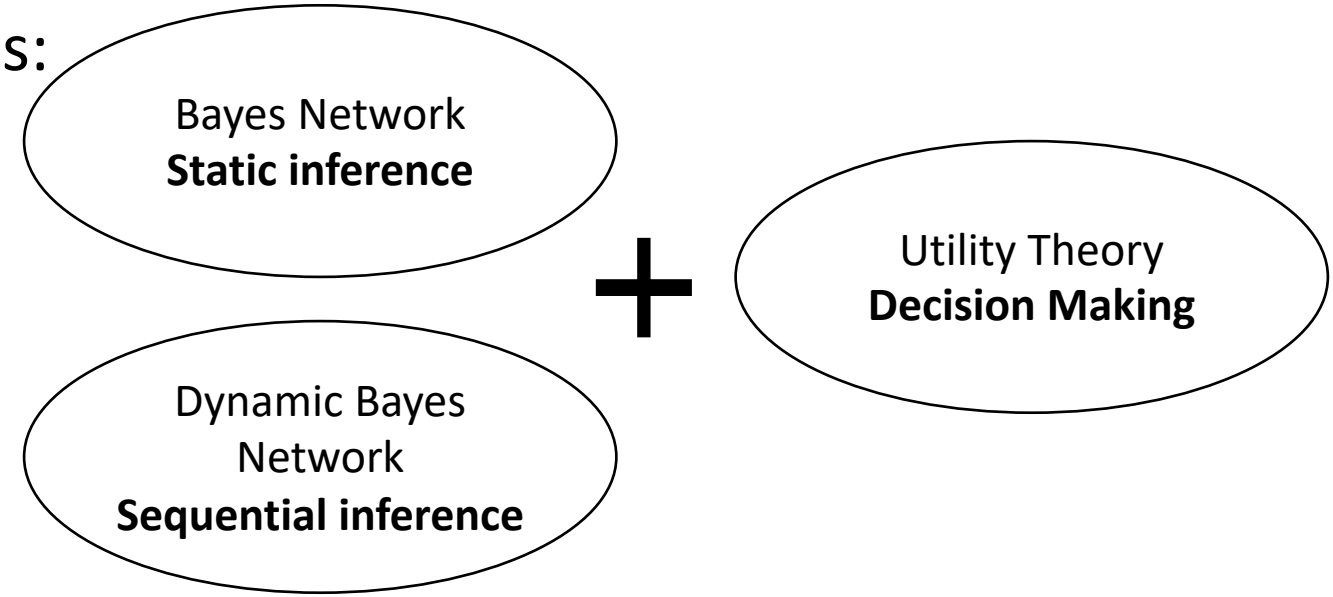
Dr. Bowen Hui
Computer Science
University of British Columbia Okanagan

Plan for Upcoming Lectures

- Lecture 1:
  - Regression -> classification -> Bayesian networks
- Lecture 2:
  - Rational decision making
- Lecture 3 (this class):
  - Sequential decision making

Single Agent Decision Making

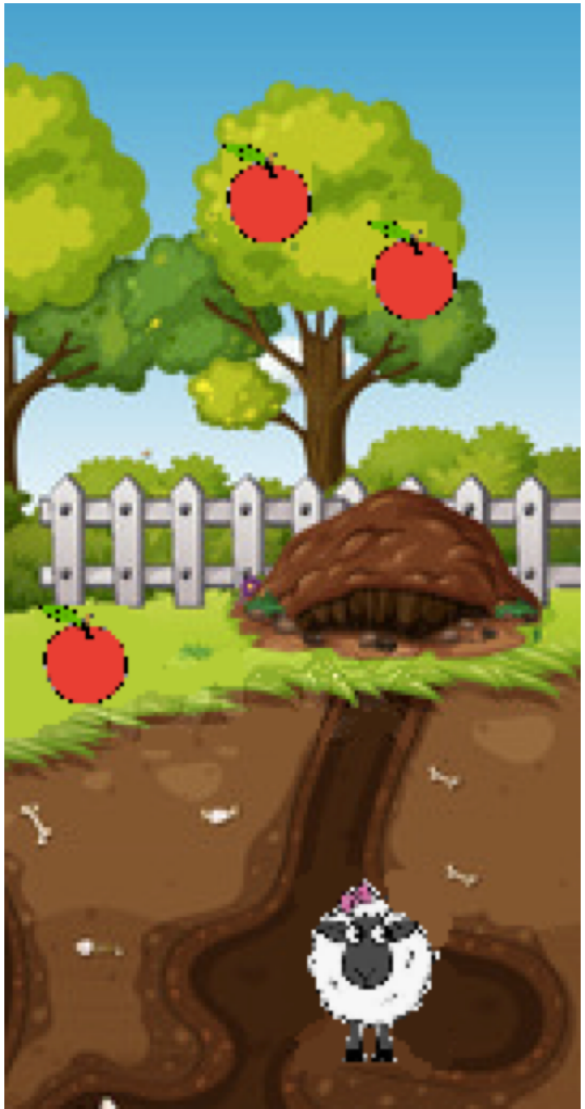
- Earlier classes:



- Markov decision process (MDP)
  - Sequential decision making (under certainty)
- Partially observable Markov decision process (POMDP)
  - Sequential decision making under uncertainty

One Shot vs. Sequential Decision Making

- What should the hungry coyote do?
  - Jump into hole and eat the sheep
  - Eat the apple

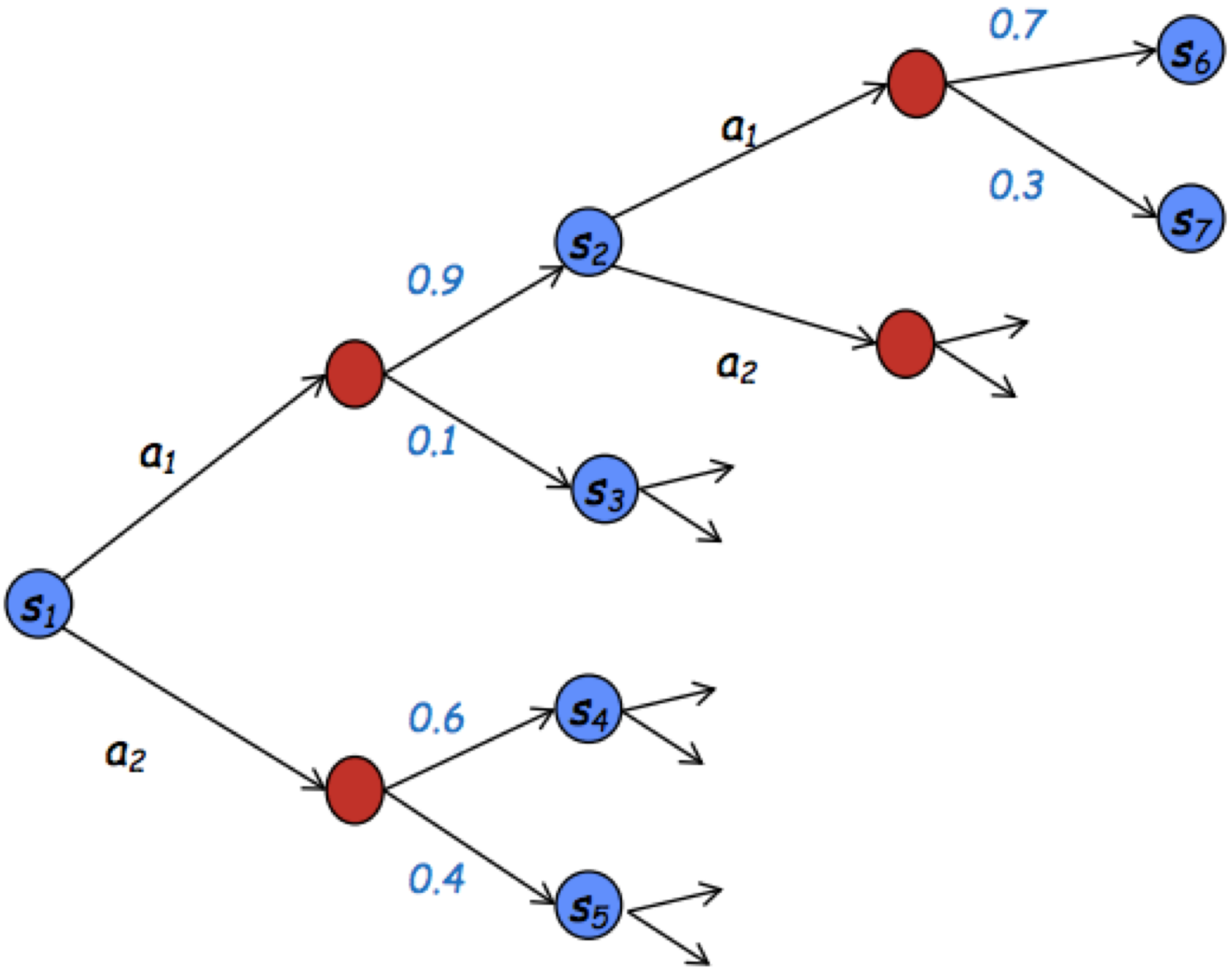


Images taken from rockysmith.net, kisspng.com, istockphoto.com, shutterstock.com

Factors to Consider

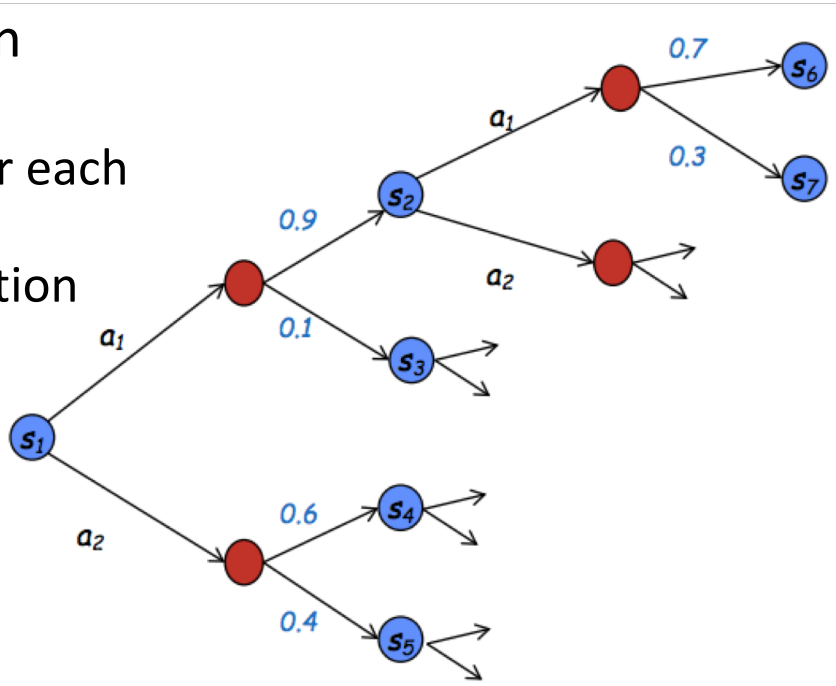
- In real life, we take actions because:
  - They give us immediate benefits
  - They lead to other opportunities (rewards we get in future steps)
  - They provide information for future decisions (separate discussion)
  - A combination of the above

A Simple Perspective



A Simple Perspective

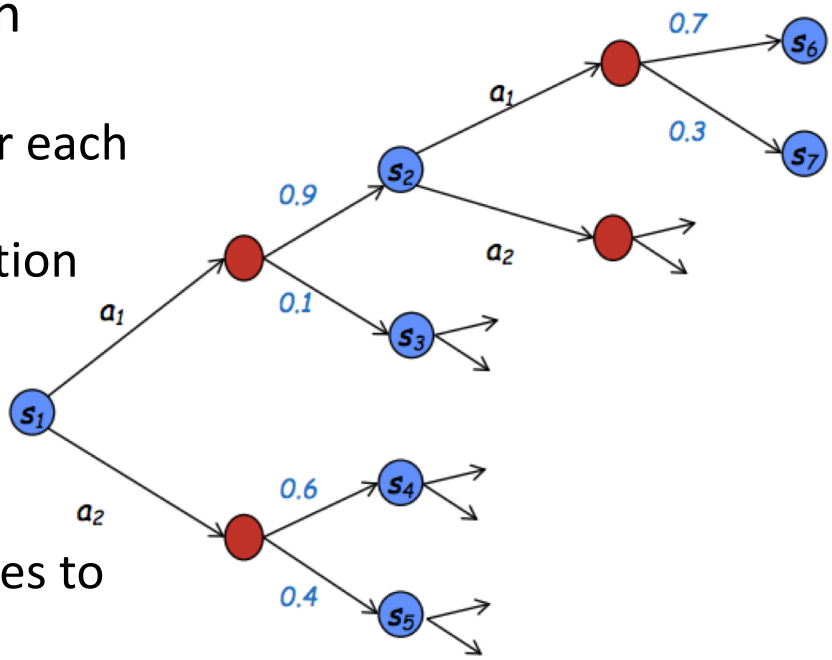
- Compute the best sequence of action
  - Assign utility to each trajectory
  - Compute probability of trajectory for each action sequence
  - Compute expected utility of each action sequence
  - Apply MEU



A Simple Perspective

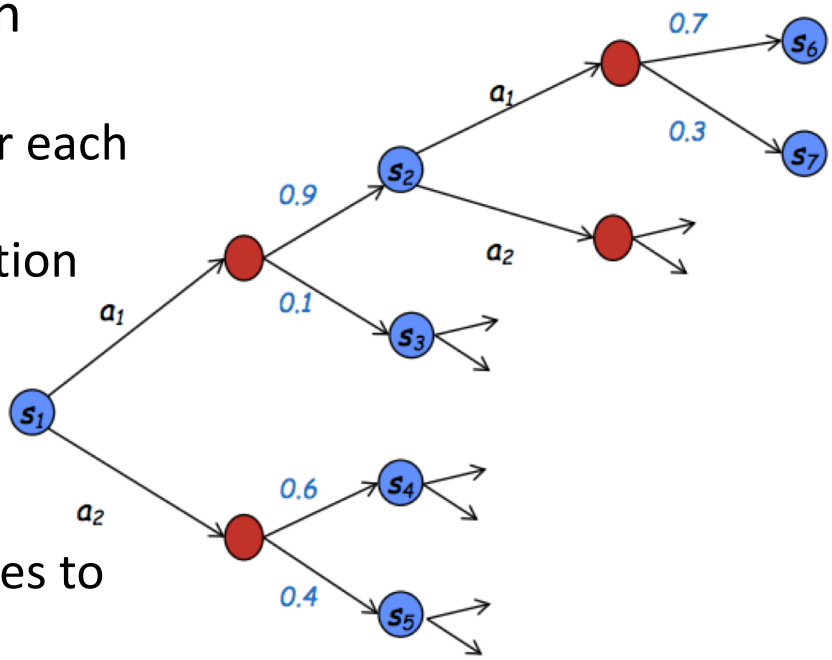
- Compute the best sequence of action
  - Assign utility to each trajectory
  - Compute probability of trajectory for each action sequence
  - Compute expected utility of each action sequence
  - Apply MEU

- Computational problems:
  - k actions, t stages: k^t action sequences to evaluate
  - n outcomes per action: $k^t n^t$ trajectories



A Simple Perspective

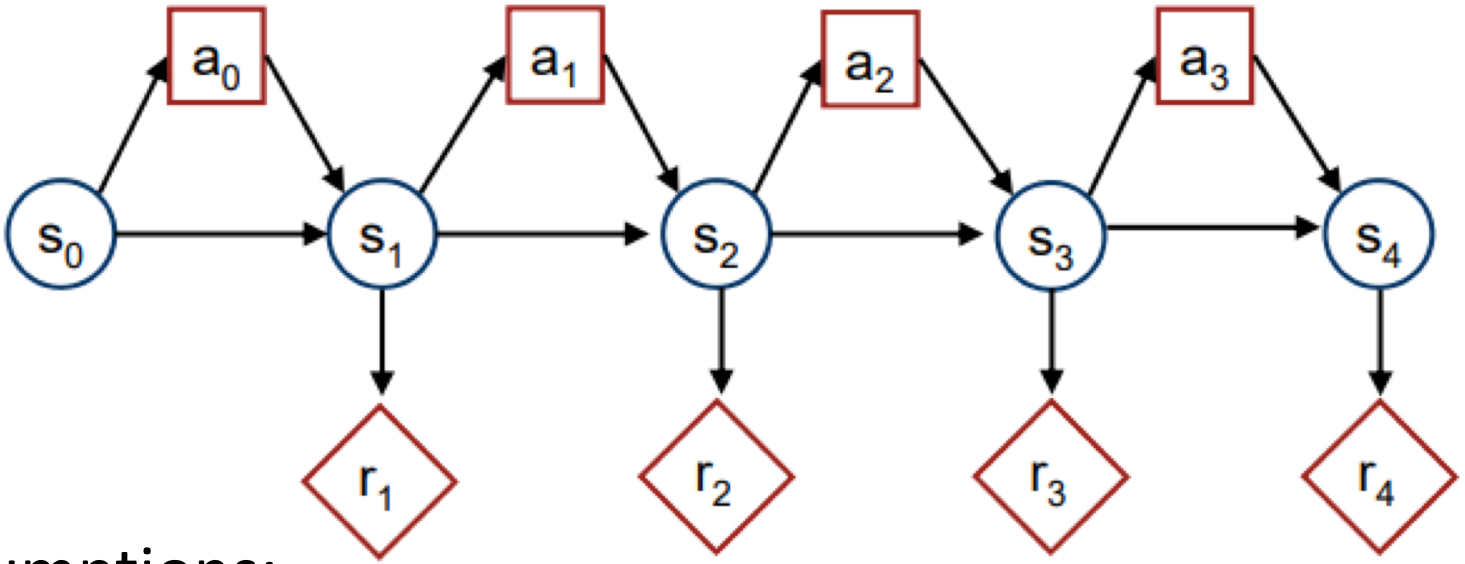
- Compute the best sequence of action
  - Assign utility to each trajectory
  - Compute probability of trajectory for each action sequence
  - Compute expected utility of each action sequence
  - Apply MEU



- Computational problems:
  - k actions, t stages: k^t action sequences to evaluate
  - n outcomes per action: $k^t n^t$ trajectories
- Conceptual problems:
  - Easier to think of utility of individual states
  - May want to do single actions rather than sequences

Intuition for MDP

- Markov process with:
  - Decision (action) nodes
  - Utility (reward) nodes



- Assumptions:
  - States are fully observable

Horizon

- Recall coyote example

- Finite horizon
  - Best action is different at each time step
  - Optimal policy is non-stationary

- Infinite horizon
  - Same best action at each time step
  - Optimal policy is stationary

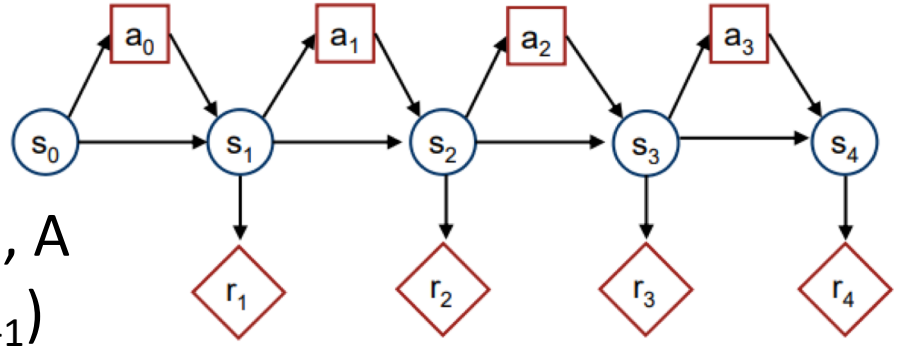
Discounted/Average Rewards

- If process is infinite, isn't $\Sigma_t R(s_t)$ infinite?

- Solution 1: discounted rewards
  - Introduce discount factor, γ
  - Finite utility: $\sum_t \gamma^t R(s_t)$ is a geometric sum
  - γ is like an inflation rate of $\frac{1}{\gamma - 1}$
  - Intuition: prefer reward sooner rather than later

- Solution 2: average rewards
  - More complicated computationally

Markov Decision Process (MDP)

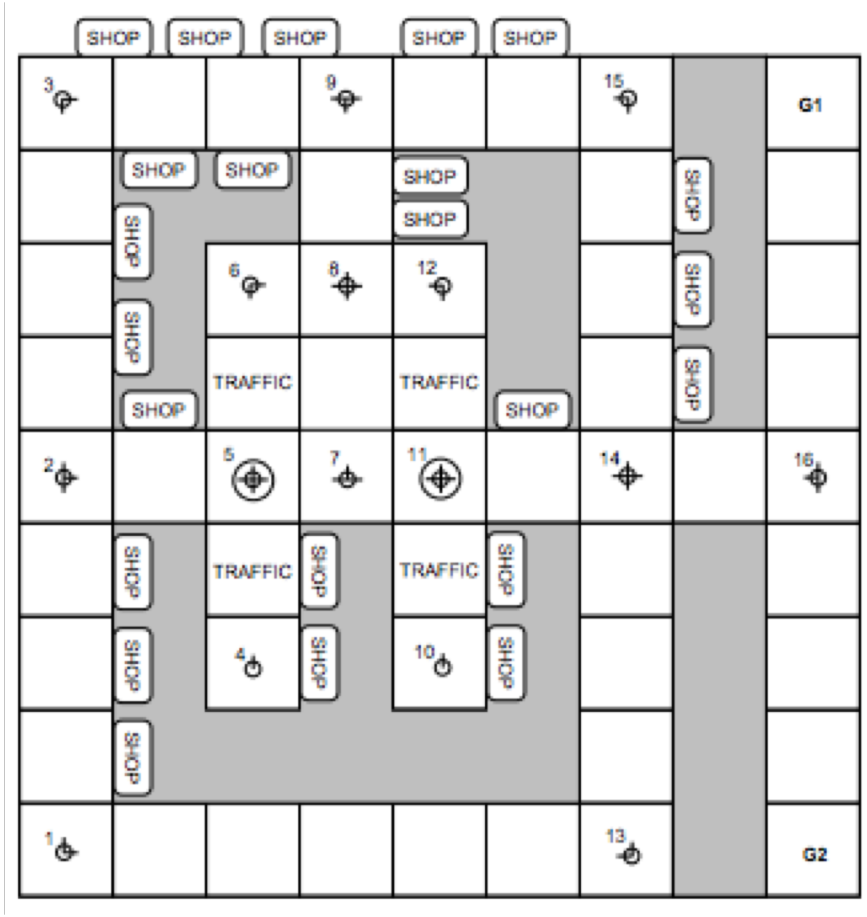


- Definition
  - Set of states, S
  - Set of actions (i.e., decisions), A
  - Transition model, $\Pr(s_t \mid a_{t-1}, s_{t-1})$
  - Reward model (i.e., utility), $R(s_t)$
    could have a more general reward model (see example)
  - Discount factor, $0 \leq \gamma \leq 1$
  - Horizon, h

- Goal: Solve for the optimal policy
  - A policy is a mapping from states to actions
    i.e.,$\pi(s_t) = a_t$

Airport Navigation Example

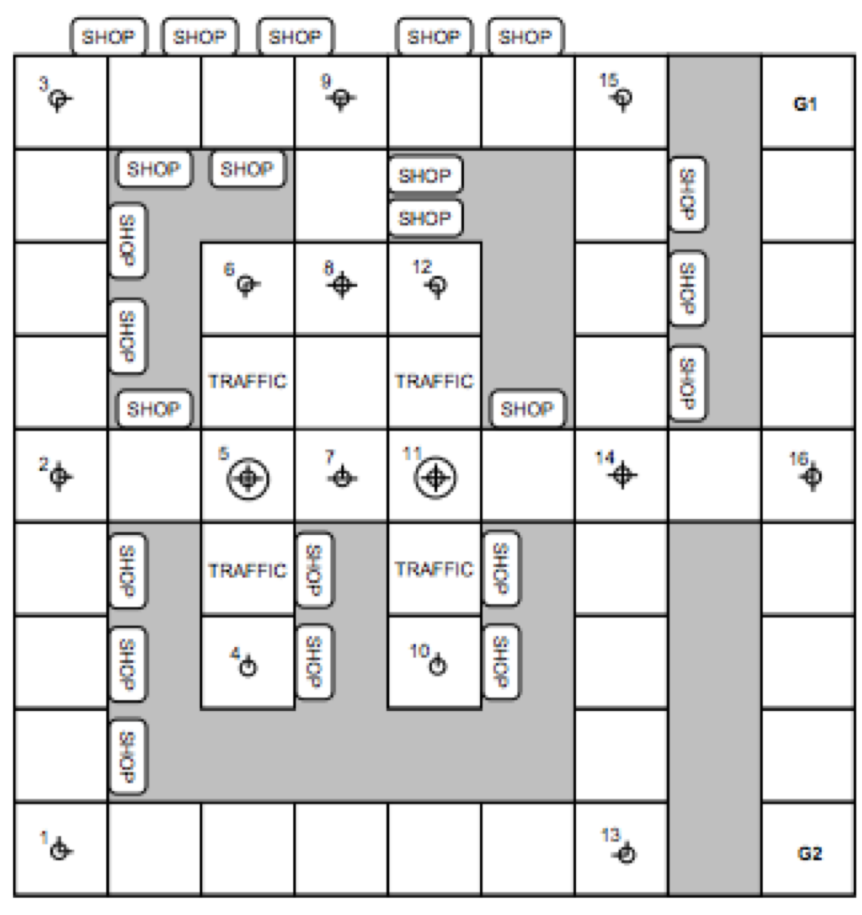
- User at airport terminal
- Goal: Go from current loc to departure gate (G1)
- System recommends an action when user is at one of 16 transmitters
- User wants to shop for souvenir
- 4 high traffic, timely areas



Bohnenberger & Jameson. When policies are better than plans: Decision-theoretic planning of recommendation sequences. IUI'2001.

Airport Navigation Example

- User at airport terminal
- Goal: Go from current loc to departure gate (G1)
- System recommends an action when user is at one of 16 transmitters
- User wants to shop for souvenir
- 4 high traffic, timely areas
- 2 “danger” zones where user might take wrong turn
- System suggestions:
  - Suggestion via speech, moves user faster
  - Suggestion via map, user is more likely to find gift



Bohnenberger & Jameson. When policies are better than plans: Decision-theoretic planning of recommendation sequences. IUI’2001.

Example MDP Specification

- State features:
  - Current user location (one of the transmitters)
  - Gift has been bought or not
- Actions:
  - Speech suggestion
  - Map suggestion
- Transition model, $Pr(s_t|s_{t-1},a_{t-1})$:
  - Probabilities from state and action to next state

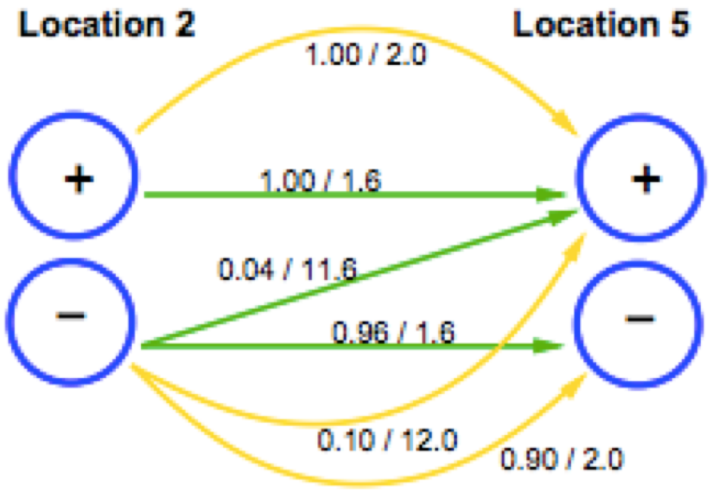


Figure 2. State transitions, probabilities, and costs associated with a recommendation that $\mathcal{U}$ go from Location 2 to Location 5.

(Dark and light arrows represent possible transitions when $\mathcal{S}$ uses speech mode and map mode, respectively. Each pair of numbers next to an arrow gives (a) the probability that the corresponding transition will be made if the recommendation is given; and (b) the cost of making that transition.)

Example MDP Specification

- State features:
  - Current user location (one of the transmitters)
  - Gift has been bought or not
- Actions:
  - Speech suggestion
  - Map suggestion
- Transition model, $Pr(s_t|s_{t-1},a_{t-1})$:
  - Probabilities from state and action to next state
- Reward model, $R(s_t|s_{t-1},a_{t-1})$:
  - Time to go from one state to the next
  - Arriving at departure gate with no gift is 0
  - Arriving at departure gate with gift is some non-negative number to reflect importance of gift relative to arriving as early as possible

Policy Optimization

- Policy evaluation:
  - Compute expected utility

$$EU(\pi) = \sum_{t=0}^{h} \gamma^t \sum_{s_t} \Pr(s_t|\pi)\, R(s_t)$$

  - Optimal policy π^* is one with highest EU
    i.e., $EU(\pi) \leq EU(\pi^*)$, for all π

- Algorithms to optimize policy:
  - Value iteration (equivalent to variable elimination)
  - Policy iteration
  - Linear programming

Value Iteration

- Performs dynamic programming
- Value when no time left:
  $V(s_h) = R(s_h)$
- Value with 1 time step left:
  $V(s_{h-1}) = \max_{a_{h-1}} R(s_{h-1}) + \gamma \sum_{s_h} \Pr(s_h | s_{h-1}, a_{h-1}) V(s_h)$
- Value with 2 time steps left:
  $V(s_{h-2}) = \max_{a_{h-2}} R(s_{h-2}) + \gamma \sum_{s_{h-1}} \Pr(s_{h-1} | s_{h-2}, a_{h-2}) V(s_{h-1})$
- etc. . . .

- Bellman's equation:
  $V(s_t) = \max_{a_t} R(s_t) + \gamma \sum_{s_{t+1}} \Pr(s_{t+1} | s_t, a_t) \; V(s_{t+1})$
- The best action:
  $a_t^* = \text{argmax}_{a_t} R(s_t) + \gamma \sum_{s_{t+1}} \Pr(s_{t+1} | s_t, a_t) \; V(s_{t+1})$

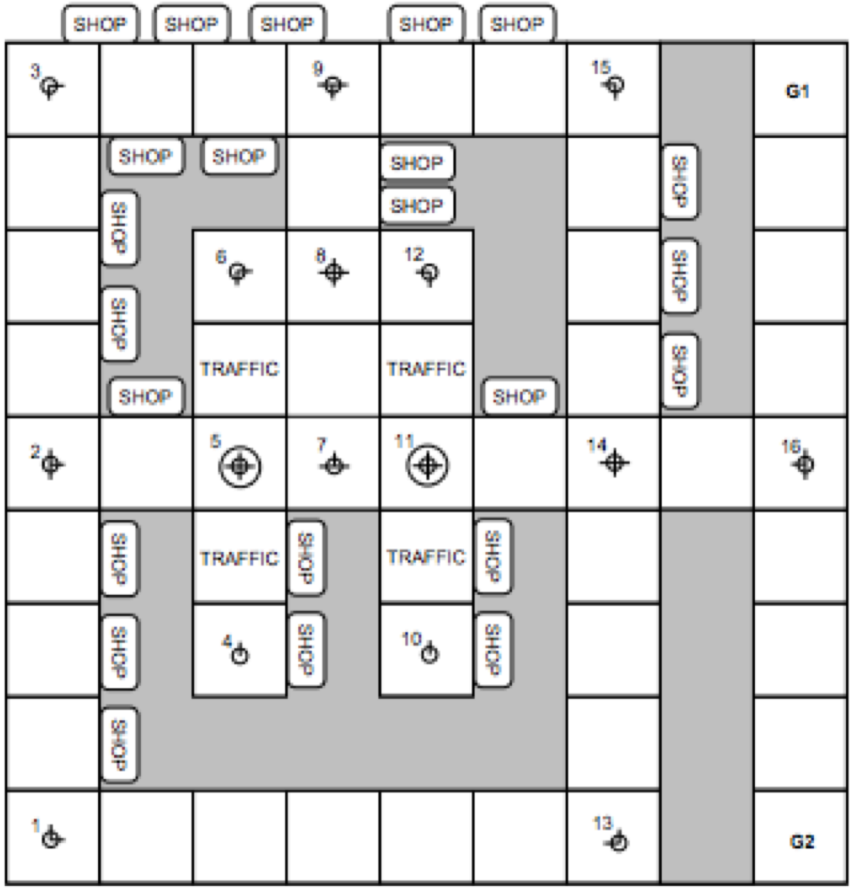
Implementing Value Iteration

- Create MDP
  - Define states and actions
  - Define transition model (one matrix for each action)
  - Define reward model (vector)
  - Define discount factor with infinite horizon

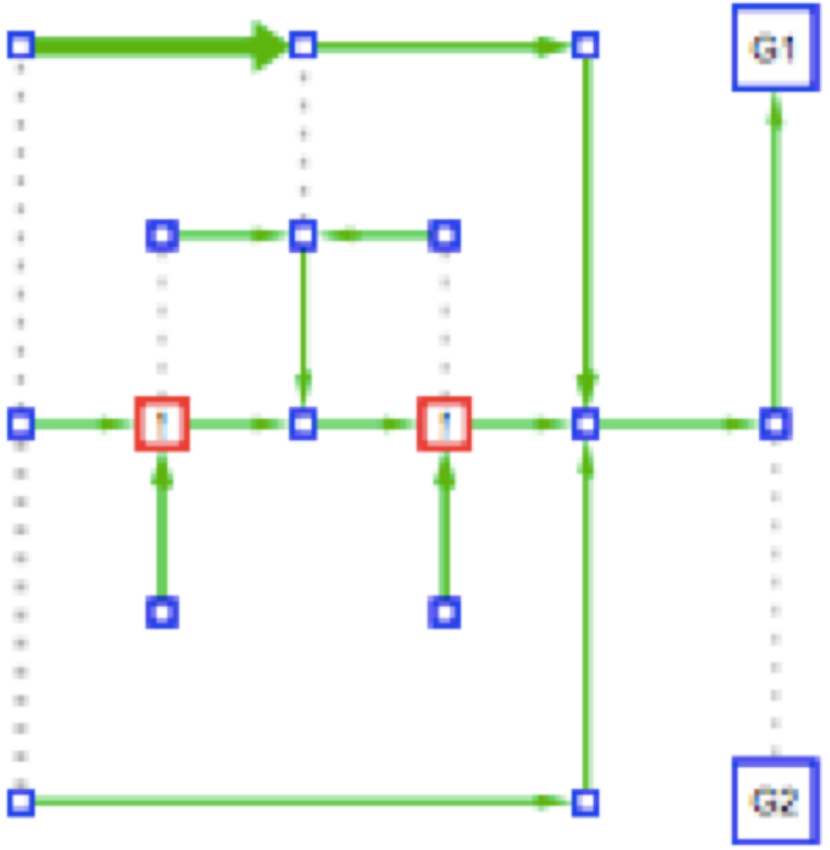
- Compute $V(s_0) = R(s)$
- For t=1, 2, …, repeat:
  - Compute $V(s_t) = \max_{a_t} R(s_t) + \gamma \sum_{s_{t+1}} \Pr(s_{t+1}|s_t, a_t)\ V(s_{t+1})$
  - Compute $\pi^*(s_t) = \text{argmax}_{a_t} R(s_t) + \gamma \sum_{s_{t+1}} \Pr(s_{t+1}|s_t, a_t)\, V(s_{t+1})$
  - If $|V(s_t) - V(s_{t-1})| \leq \varepsilon$ then stop

- Return π^*

Example Solved Policy

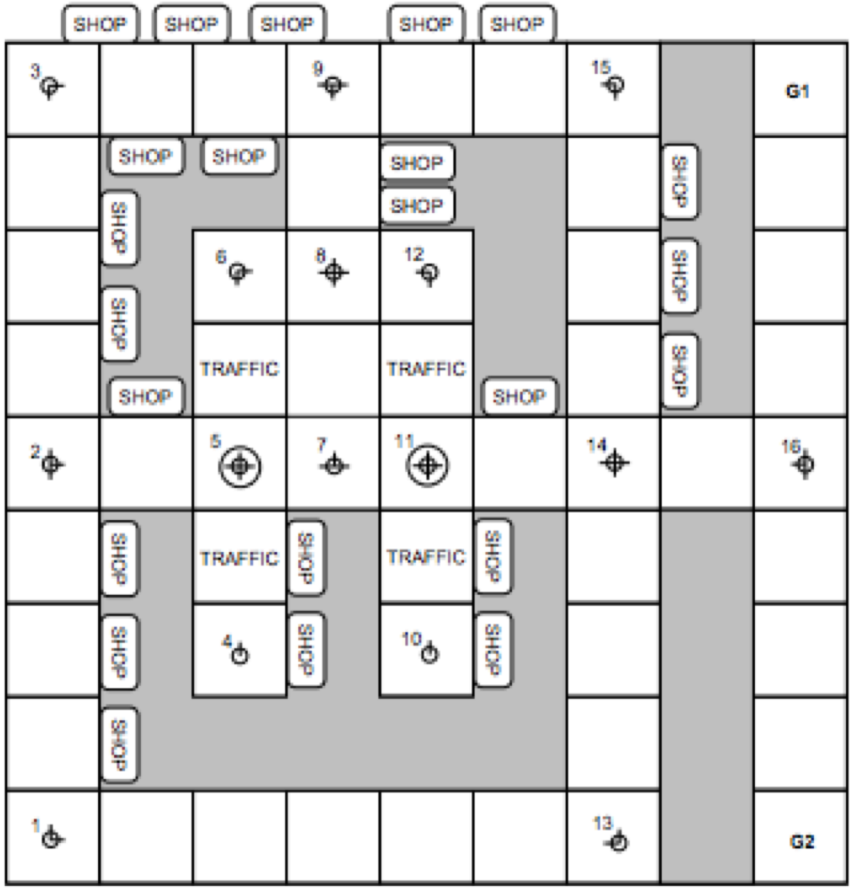


Original map

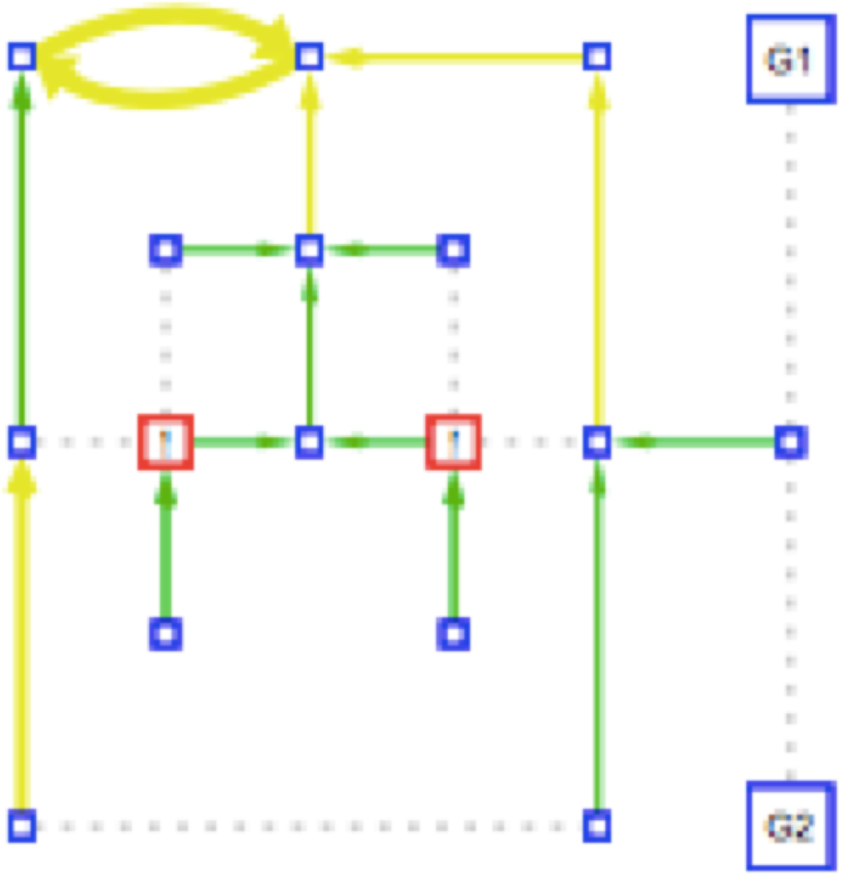


Policy when gift is of no importance or already found

Example Solved Policy



Original map

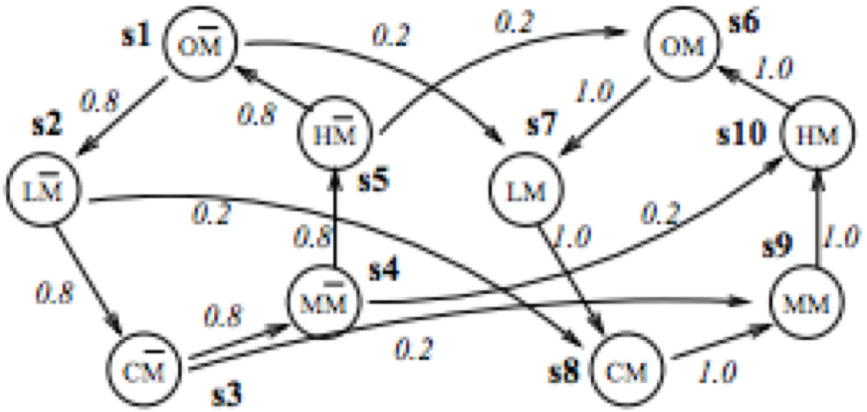


Policy when gift is of high importance

Flat vs. Factored Representation

- Flat representation:
  - States have an explicit representation which are directly enumerated
- Factored representation:
  - More compact representation describes states in terms of features
    - Features: loc = { 1, 2, 3 }, gift = { T, F }
    - Then, states = { 1T, 1F, 2T, 2F, 3T, 3F }
  - Often more convenient to model
  - Exploits structure

Comparison: Flat Representation

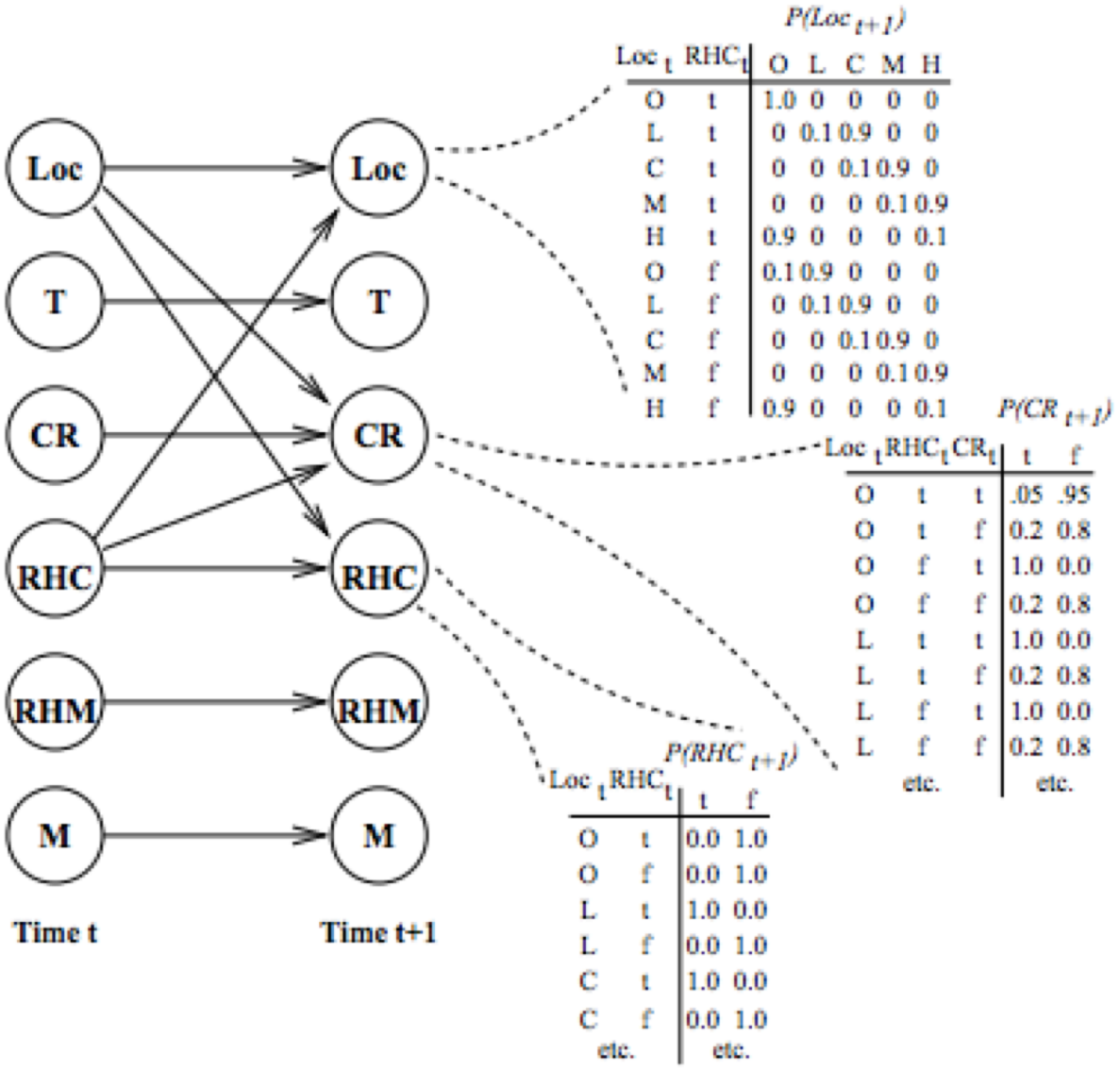


	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
s_1	0.0	0.8	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0
s_2	0.0	0.0	0.8	0.0	0.0	0.0	0.0	0.2	0.0	0.0
s_3	0.0	0.0	0.0	0.8	0.0	0.0	0.0	0.0	0.2	0.0
s_4	0.0	0.0	0.0	0.0	0.8	0.0	0.0	0.0	0.0	0.2
s_5	0.8	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0
s_6	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
s_7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
s_8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
s_9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
s_{10}	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0

Figure 5: The state-transition diagram and transition matrix for a moving robot.

Taken from (Boutilier, Dean, Hanks, 1999)

Comparison: Factored Representation



Can "flatten" out
If needed

Taken from (Boutilier, Dean, Hanks, 1999)

MDP Modeling Example

- Suggest promotion to friend X or friend Y?

MDP Modeling Example

- Suggest promotion to friend X or friend Y?
  - States:
    Spread trend to friends, friend buys similar products, product reputation, company loyalty

MDP Modeling Example

- Suggest promotion to friend X or friend Y?
  - States:
    Spread trend to friends, friend buys similar products, product reputation, company loyalty
  - Transition model:
    Pr(Spread' | A, Spread)
    Pr(BuySimilar' | A, Spread)
    Pr(Reputation' | A, Reputation)
    Pr(Loyalty' | A, Loyalty)

MDP Modeling Example

- Suggest promotion to friend X or friend Y?
  - States:
    Spread trend to friends, friend buys similar products, product reputation, company loyalty
  - Transition model:
    Pr(Spread' | A, Spread)
    Pr(BuySimilar' | A, Spread)
    Pr(Reputation' | A, Reputation)
    Pr(Loyalty' | A, Loyalty)
  - Reward model:
    Reward(BuySimilar, Reputation, Loyalty)

MDP Modeling Example

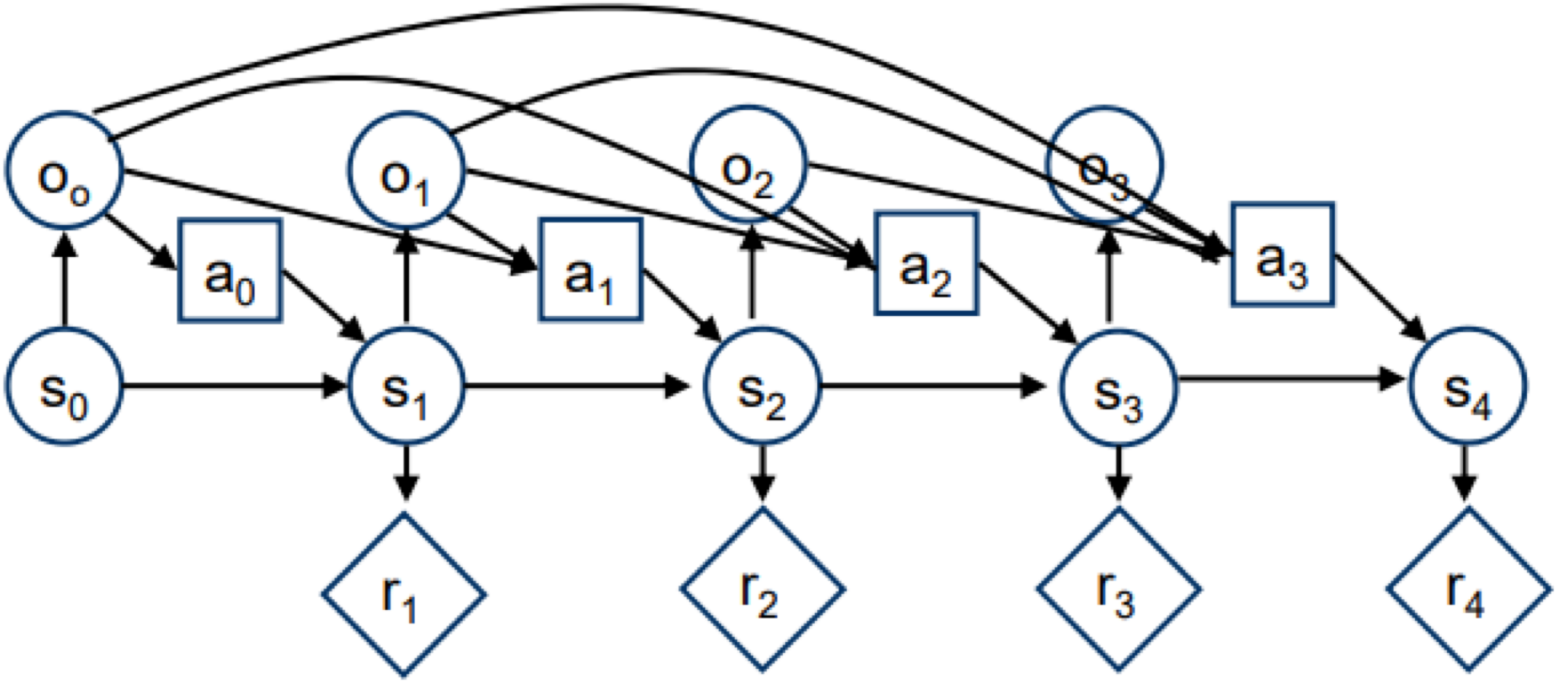
- Suggest promotion to friend X or friend Y?
  - States:
    Spread trend to friends, friend buys similar products, product reputation, company loyalty
  - Transition model:
    Pr(Spread' | A, Spread)
    Pr(BuySimilar' | A, Spread)
    Pr(Reputation' | A, Reputation)
    Pr(Loyalty' | A, Loyalty)
  - Reward model:
    Reward(BuySimilar, Reputation, Loyalty)

How to populate transition model?

How to define reward model?

POMDPs

- Realistically, states are typically not fully observable
  - POMDPs are more general and harder to solve than MDPs
  - Added set of observations, O, and an observation model, $Pr(o_t|s_t)$
  - Policy: mapping from past observations to actions



Key Ideas

- Main concept
  - One shot vs. sequential decision making
- Representation:
  - MDPs assume states are fully observable and 1st order Markov
  - Policies are stationary
  - Flat vs. factored representation of states
  - POMDPs have added observations and observation model
- Main task of interest:
  - Solve for optimal policy with highest expected utility
- Algorithm:
  - Value iteration makes use of dynamic programming

Further Readings

- C. Boutilier, T. Dean, and S. Hanks. Decision Theoretic Planning: Structural Assumptions and Computational Leverage. Journal of Artificial Intelligence Research, 11:1—94, 1999.

- L.P. Kaelbling, M.L. Littman, A.R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. Artificial Intelligence, 101:1-2, pages 99—134, 1998.