COLLEGE OF ENGINEERING
INDUSTRIAL & OPERATIONS ENGINEERING
UNIVERSITY OF MICHIGAN

Sequential Decision Making in Medicine:

MDPs, POMDPs, and Beyond…

CHOIR Healthcare Operations Research Summer School
University of Twente
July 23, 2017

Brian Denton
University of Michigan

Goals of this Tutorial

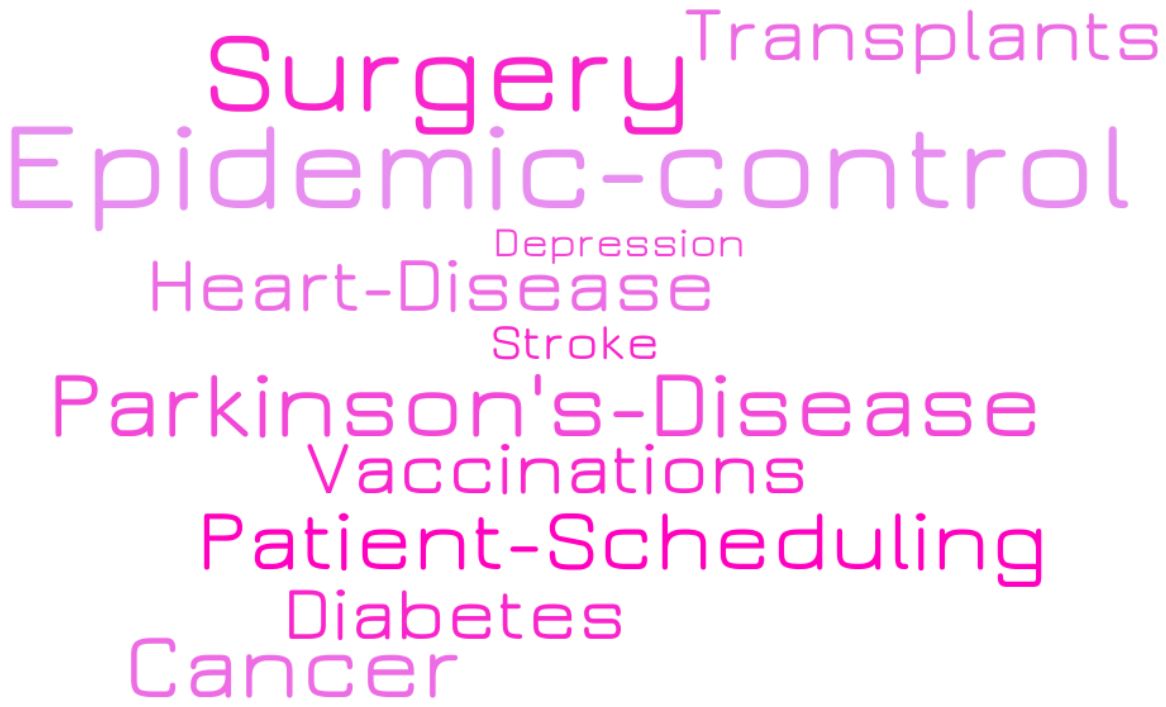
- To help you develop some understanding of the theory and mathematical methods associated with sequential decision making
- To illustrate the use of these methods in healthcare
- To convince you to drop everything and focus on problems involving sequential decision making!

Sections

- Finite Horizon MDPs
- Infinite Horizon MDPs
- Special Topics
  - Partially observable MDPs
  - Robust MDPs
  - Approximate DP
  - Reinforcement Learning

Warning: This tutorial contains more information about sequential decision making than can be safely ingested in 2.5 hours.

Healthcare Problems Addressed by MDPs



Schaefer et al. 2005. "Modeling medical treatment using Markov decision processes." In *Operations research and health care*, pp. 593-612. Springer US, 2005.

Dıez, et al. 2011 "MDPs in medicine: opportunities and challenges." In *Decision Making in Partially Observable, Uncertain Worlds: Exploring Insights from Multiple Communities (IJCAI Workshop), Barcelona (Spain)*, vol. 9, p. 14. 2011.

Steimle and Denton. 2017. "Markov decision processes for screening and treatment of chronic diseases." In *Markov Decision Processes in Practice*, pp. 189-222. Springer International Publishing, 2017.

Dynamic programming (DP) has been used to solve many optimization problems

- Often, but not always involving the element of time

In many applications DP obtains solutions by:

- Working backward from the end of the problem toward the beginning

- Breaking up a large problem into a series of smaller, easier problems

Richard Bellman

Dynamic programming (DP) dates back to early work of **Richard Bellman** in the 1940's

1954 Paper by Bellman describes the foundation for DP

Since its development DP has been applied to fields of mathematics, engineering, biology, chemistry, and many others

For more history on Richard Bellman see: http://www.gap-system.org/~history/Biographies/Bellman.html

Definitions

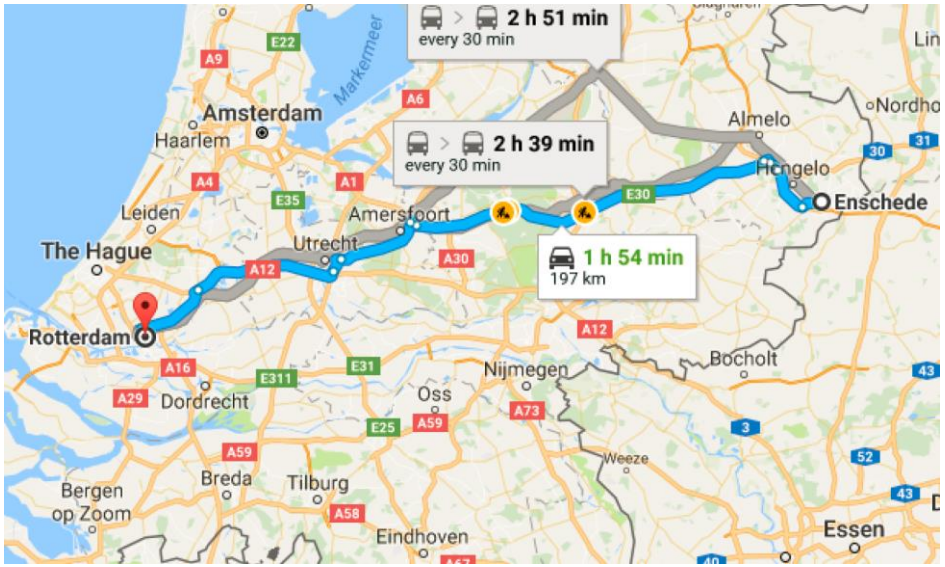
A **policy** defines the **action** to take in each possible **state** of the system

An **optimal policy** defines the **optimal action** to take for each **state** that will achieve some goal such as

- Maximize rewards gained over time
- Minimize costs paid over time
- Achieve an outcome with high probability

Shortest Path Problems

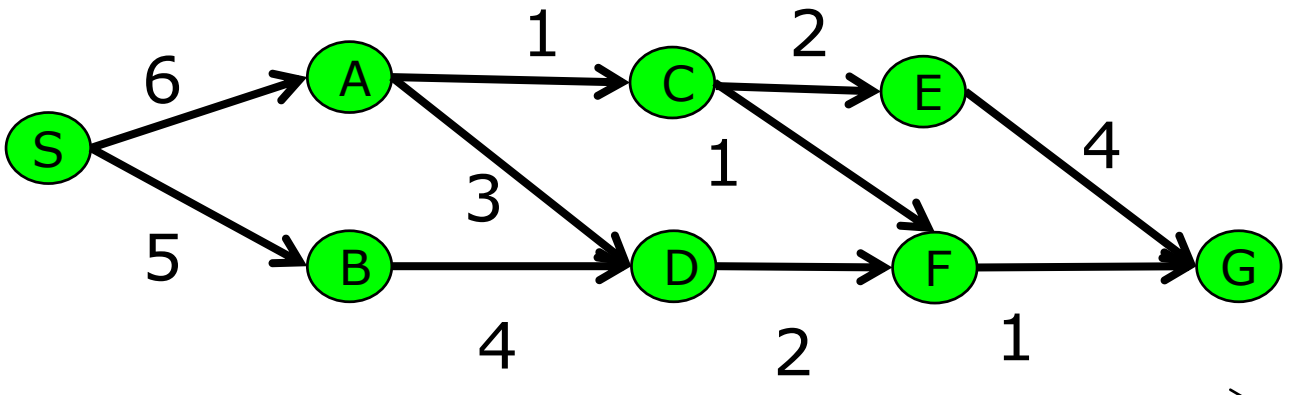
- DPs can be used for finding the shortest path that joins two points in a network



- Many problems can be formulated as a shortest path problem

Shortest Path Example

What is the shortest path in this directed graph?



Dynamic Program Formulation

Main Elements

- States: vertices of the graph
- Transfer Function: edges of the graph
- Actions: which vertex to move to
- Rewards: cost associated with selecting an edge

Goal: Starting from vertex S, select the action at each vertex that will minimize total edge distance travelled to reach vertex G

General Dynamic Program Formulation

Let a DP have **states**, $s_t \in S$, **actions** $a_t \in A$, **rewards**, $r_t(s_t, a_t)$, and an **optimal value function**, $v_t(s_t)$, defined for stages $t = 1, 2, \ldots, T$

Optimality Equations:

$$v_t(s_t) = \max_{a_t \in A} \{r_t(s_t, a_t) + v_{t+1}(s_{t+1})\}, \qquad \forall s_t$$

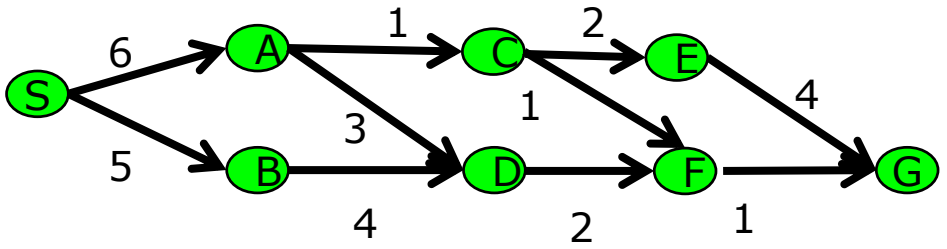
$$v_T(s_T) = R(s_T), \qquad \forall s_T$$

$v_t(s_t)$ is the maximum total reward for all stages $t, t+1, \ldots, T_t$, also called the "optimal value to go"

Transition from s_t to s_{t+1} governed by a **transfer equation**:

$$s_{t+1} = g(s_t, a_t)$$

Computing the Optimal Policy



- Start by finding the stage 5 distance:

  $v_5(G)=0$

- Stage 4: compute shortest path from E and F:

  from E→ G and F→ G: $v_4(E) = 4+0$, $v_4(F) = 1+0$

- Stage 3: compute the shortest path from C and D:

  $v_3(C) = \min\{2+V_4(E), 1+V_4(F)\}=2$, $v_3(D) = 2+ v_4(F) = 3$

- Stage 2: compute the shortest path from A and B:

  $v_2(A) = \min\{1+v_3(C), 3+v_3(D)\} = 3$, $v_2(B) = 4+v_3(D) = 7$

- Stage 1: compute the shortest path from S:

  $v_1(S) = \min\{6+v_2(A), 5+v_2(B)\} = 9$

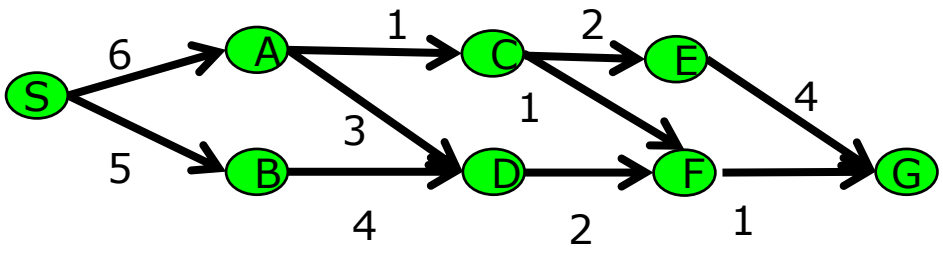
Shortest path: S→A→C→F→G

Question: What's the difference between the optimal policy and shortest path?

Optimal Policy

Policies can be expressed as "**lookup tables**".
The lookup table for this shortest path problem is:

State	Action
S	Move to A
A	Move to C
B	Move to D
C	Move to F
D	Move to F
E	Move to G
F	Move to G
G	∅



Principle of Optimality

Following is a quote from a 1954 paper by Richard Bellman:

“An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.”

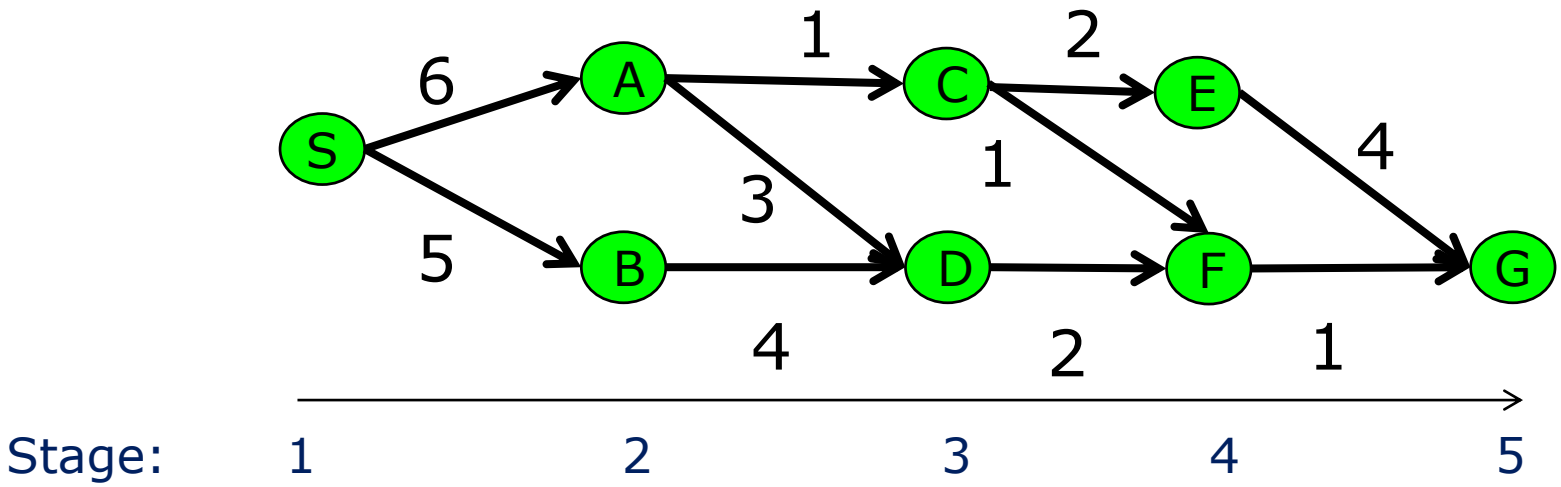
Bellman, 1954. “The Theory of Dynamic Programming,” Bulletin of the American Mathematics Society, 60(6), 503-515

Principle of Optimality

Principle of Optimality: Any “subpath” in the optimal path must be the shortest path between the two vertices

To find the best action from node S we need to compare

- Distance from S to A + Shortest Path from A to G
- Distance from S to B + Shortest Path from B to G



What About Uncertainty?

Uncertainty arises in many ways in health care:

- Health status
- Treatment effects
- Diagnostic test results
- Patient demand for services
- Patient “no-shows”

The first and easiest step to address uncertainty is a **Markov decision process (MDP)**

Assumptions

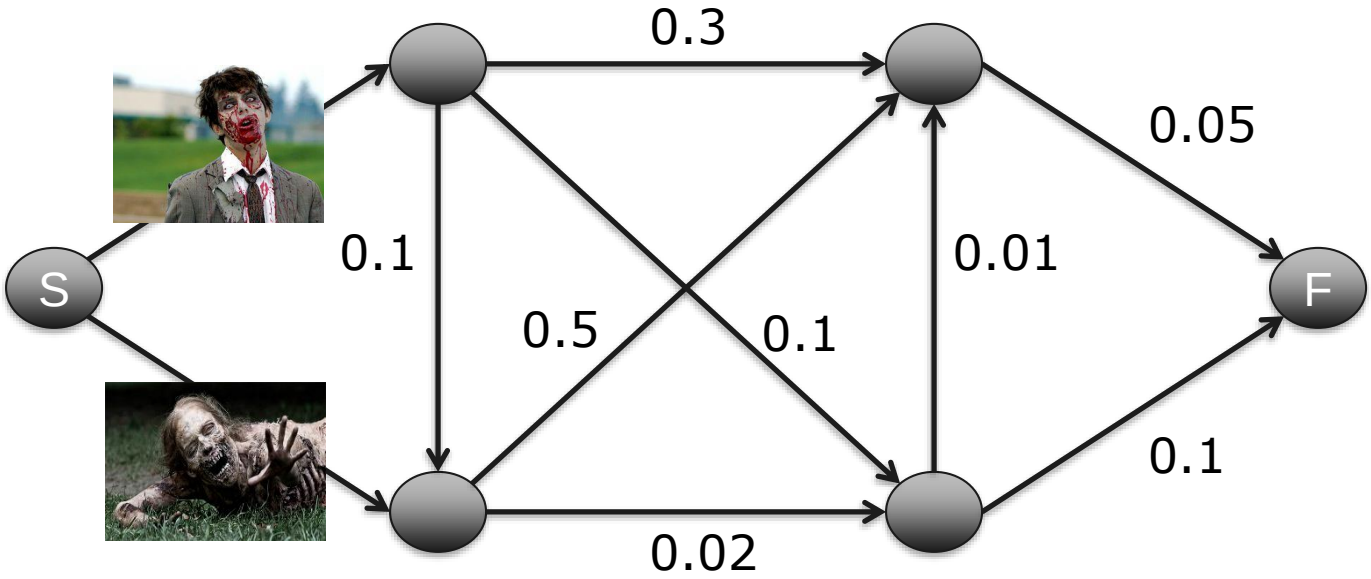
For this tutorial we will make the following assumptions:

- The set of actions, A, and states, S, are finite
- The decision maker's goal can be represented by linear additive rewards

Zombies – A Public Health Crisis!

Formulate a DP for the following problem:

- You must traverse the following network in which edge weights are the probability of encountering a zombie
- Your goal is to find the path that minimizes the probability of encounters



Markov Chains

A discrete-time stochastic process is a **Markov chain** if, for t = 0,1,2…, and for all states

$$P(X_{t+1} = i_{t+1} | X_t = i_t, X_{t-1} = i_{t-1}, \dots, X_1 = i_1, X_0 = i_0)$$

$$= P(X_{t+1} = i_{t+1} | X_t = i_t)$$

The probability distribution of the state at time t+1 depends only on the state at time t

The vector $\mathrm{q} = (\mathrm{q}_1, \mathrm{q}_2, \dots, \mathrm{q}_\mathrm{N})$ is the **initial probability distribution** for the Markov chain at time 0

$$P(X_0 = i\ ) = q_i$$

Transition Probabilities

$P(X_{t+1} = j | X_t = i) = p_{ij}^t$ as the **transition probability**

The transition probability matrix for an N-state stationary Markov chain is:

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \vdots & \vdots & & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{bmatrix}$$

Note: Transition probability matrices for MDPs depend on the policy through the action at each decision epoch.

Induced Stochastic Process

- A policy **induces** a particular stochastic process. In a finite horizon MDP the set of possible sample paths is

$$\Omega = S \times A \times S \times A \times \cdots \times A \times S = \{S \times A\}^{N-1} \times S$$

- A **sample path,** $\omega \in \Omega$, defines the states and actions for a realization of the induced stochastic process

$$\omega = (s_1, a_1, s_2, a_2, \ldots, \mathrm{a_{N-1}}, \mathrm{s_N})$$

that occurs with probability $\mathrm{p}(\omega)$

Forward Perspective

- Let $\mathrm{p}(\omega)$ denote the probability of a sample path. Expected rewards can be written as follows:

$$W(\omega) = \sum_{t=1}^{N-1} r_t(s_t, a_t) + r_N(s_N)$$

$$E_{\omega}^{\pi}[W(\omega)] = \sum_{\omega=1}^{|\Omega|} p(\omega) W(\omega)$$

- Optimal Policy:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}}\, E_{\omega}^{\pi}[W(\omega)]$$

Optimality Equations

For all states, s_t, and all time periods, $t = 1, \ldots, T-1$

$$v_t(s_t) = \max_{a_t \in A} \left\{ r_t(s_t, a_t) + \lambda \sum_{s_{t+1} \in S} p(s_{t+1}|s_t, a_t)\, v_{t+1}(s_{t+1}) \right\}$$

Immediate Reward

Future "value to go"

Boundary Condition: $v_N(s_N) = R(s_N), \quad \forall s_N$

Fundamental Result

Theorem: Suppose $\mathrm{v_t}(s_t), \text{for all } t \text{ and } s_t$ is a solution to the optimality equations, then $v_t(s_t) = {v_t}^*(s_t), \text{ for all } t \text{ and } s_t$ the associated actions define the optimal policy π^* for the MDP.

Importance: This proves solving the optimality equations yields an optimal solution to the MDP.

Reference: These results are an aggregate of results presented in chapter 4 of Puterman.

Backwards Perspective

Algorithm (Backward Induction):

1. Set $\mathrm{t} = \mathrm{N}, v_N^*(s_N) = r_N(s_N), \forall s_N$

2. $If\ t = 1\ stop, otherwise\ go\ to\ step\ 3$

3. $Substitute\ t-1\ for\ t\ and\ compute\ v_t^*(s_t), \forall s_t, as$:

$$v_t^*(s_t) = \max_{\mathrm{a}\in A} \{\mathrm{r_t}(\mathrm{s_t}, a) + \lambda \sum_{j\in S} p_t(j|s_t, a) v_{t+1}^*(j)\}$$

$$Set\ \ a_t^*(s_t) = \arg\max_{\mathrm{a}\in A} \{\mathrm{r_t}(\mathrm{s_t}, a) + \lambda \textstyle\sum_{j\in S} p_t(j|s_t, a) v_{t+1}^*(j)\}$$

$return\ to\ step\ 2.$

Special Structured Policies

Policies with a simple structure like are:

- Easier for decision makers to understand
- Easier to implement
- Easier to solve the associated MDPs

General structure of a **control limit policy**

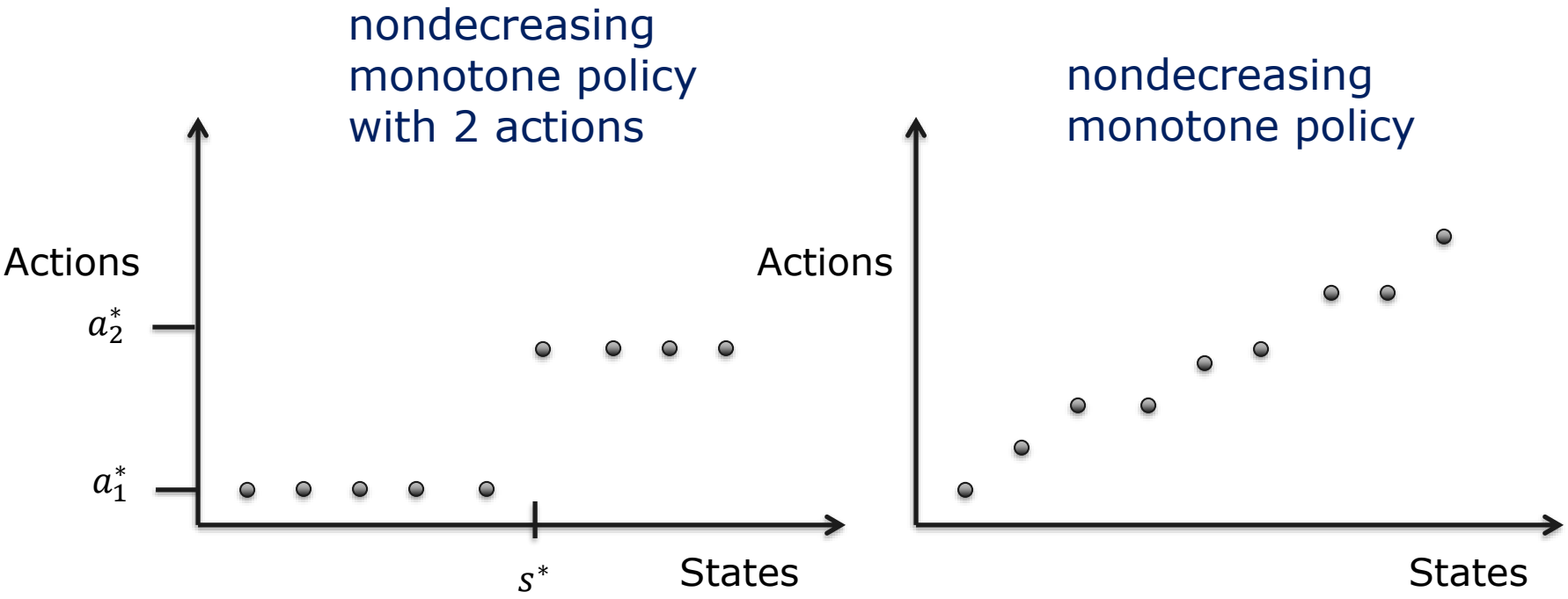
$$a_t(s_t) = \begin{cases} a_1, if\ s < s^* \\ a_2, if\ s \geq s^* \end{cases}$$

where a_1 and a_2 are alternative actions and s^* is a control limit.

Question: What conditions guarantee the existence of a control limit policy?

Monotone Policies

Definition: Control limit policies are examples of **monotone** policies. A policy is **monotone** if the *decision rule* at each stage is nonincreasing or nondecreasing with respect to the system state.



Monotonicity: Sufficient Conditions

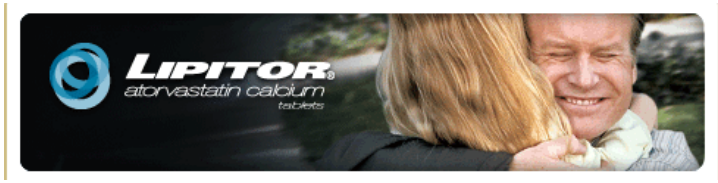
Theorem: Suppose for $t = 1, \ldots, N-1$

1. $r_t(s,a)$ is nondecreasing in s for all $a \in A$.
2. $q_t(k|s,a)$ is nondecreasing in s for all $k \in S, a \in A$.
3. $r_t(s,a)$ is **superadditive (subadditive)** on $S \times A$.
4. $q_t(k|s,a)$ is **superadditive (subadditive)** on $S \times A$, $\forall k$
5. $R_N(s)$ is nondecreasing in s.

Then there exist optimal decision rules, $d_t^*(s)$, which are nondecreasing (nonincreasing) in s for $t = 1, \ldots, N-1$.

MDP Example: Drug Treatment Initiation

Some medical treatment decisions can be viewed as a “stopping time” problem:



- Statins lower your risk of heart attack and stroke
- Treatment has side effects and cost
- Patients receive annual cholesterol tests and make a decision to:
  - initiate statins
  - defer initiation for a year

Stopping Time Problem

Optimality equations:

$$\mathrm{v_t(s)} = \max_{\mathrm{a}\in\{Q,C\}} \left\{ \mathrm{g_t(s)}, \mathrm{r(s,C)} + \sum_{\mathrm{j}\in S} p(j|s,C) v_{t+1}(j) \right\}, \forall s \in S$$

$$\mathrm{v}_N(\mathrm{s}) = \mathrm{g_N(s)}, \forall s \in S$$

States define patient health status

Action C represents decision to defer statin initiation, Q denotes decision to start statins

$\mathrm{g}_t(\mathrm{s})$ is expected survival if statins are initiated

Model Description

Stages:

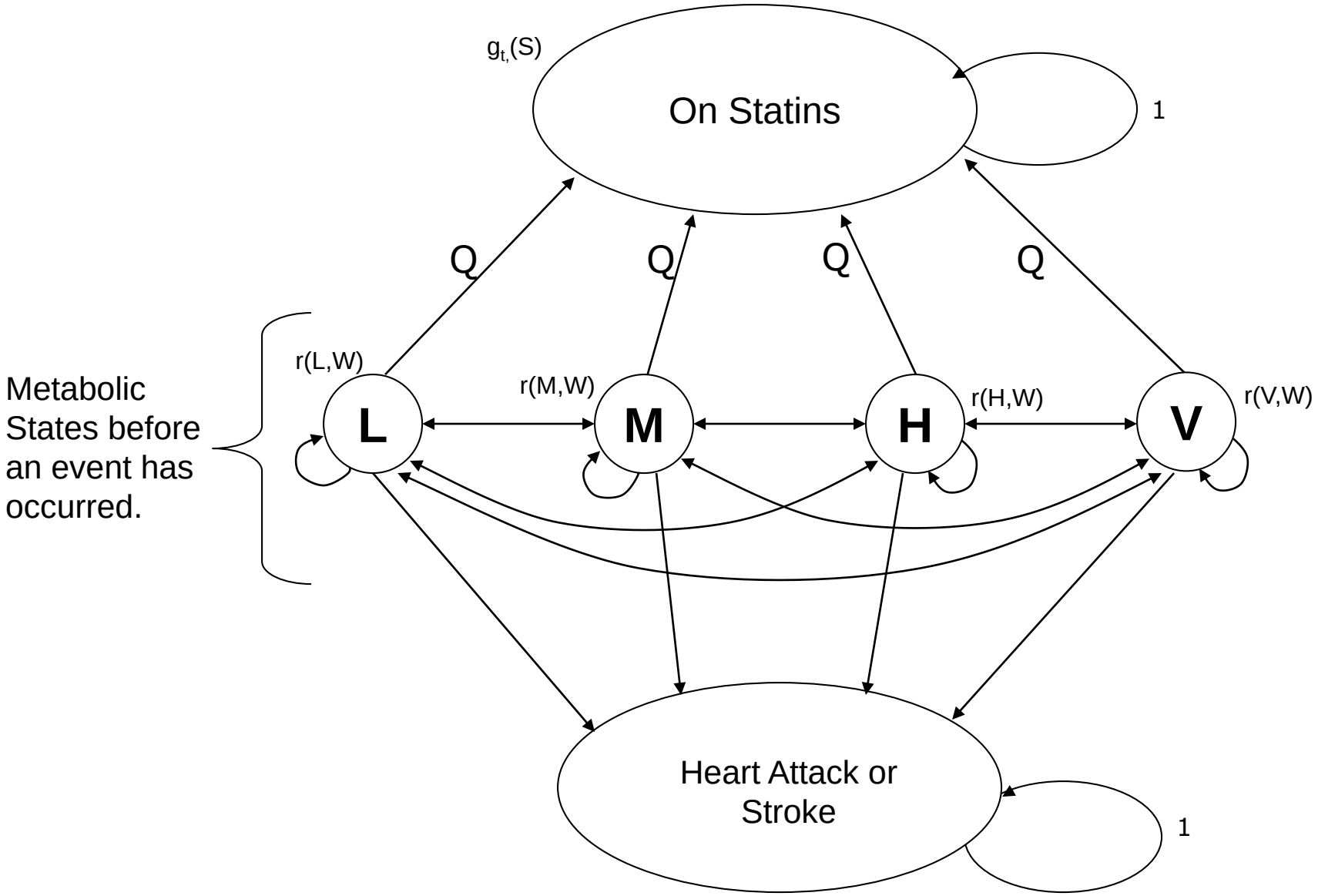
- Time horizon: Ages 40-80
- Annual decision epochs

Actions: **Initiate** (Q) or **delay** (C) statin treatment

States:

- Metabolic: Total cholesterol and HDL (each can be L, M, H, V)
- Demographic: Gender, Race, BMI, smoking status, medical history

Treatment Markov Chain



Rewards

There are various types of reward functions used in health studies like this. The simplest definition for this problem is:

- $\mathrm{r_t}\ (s_t)$ is the time between decision epochs (e.g. 1 year)

- $g_t(s_t)$ is the expected future life years adjusted for quality of life on medication

Computing Transition Probabilities

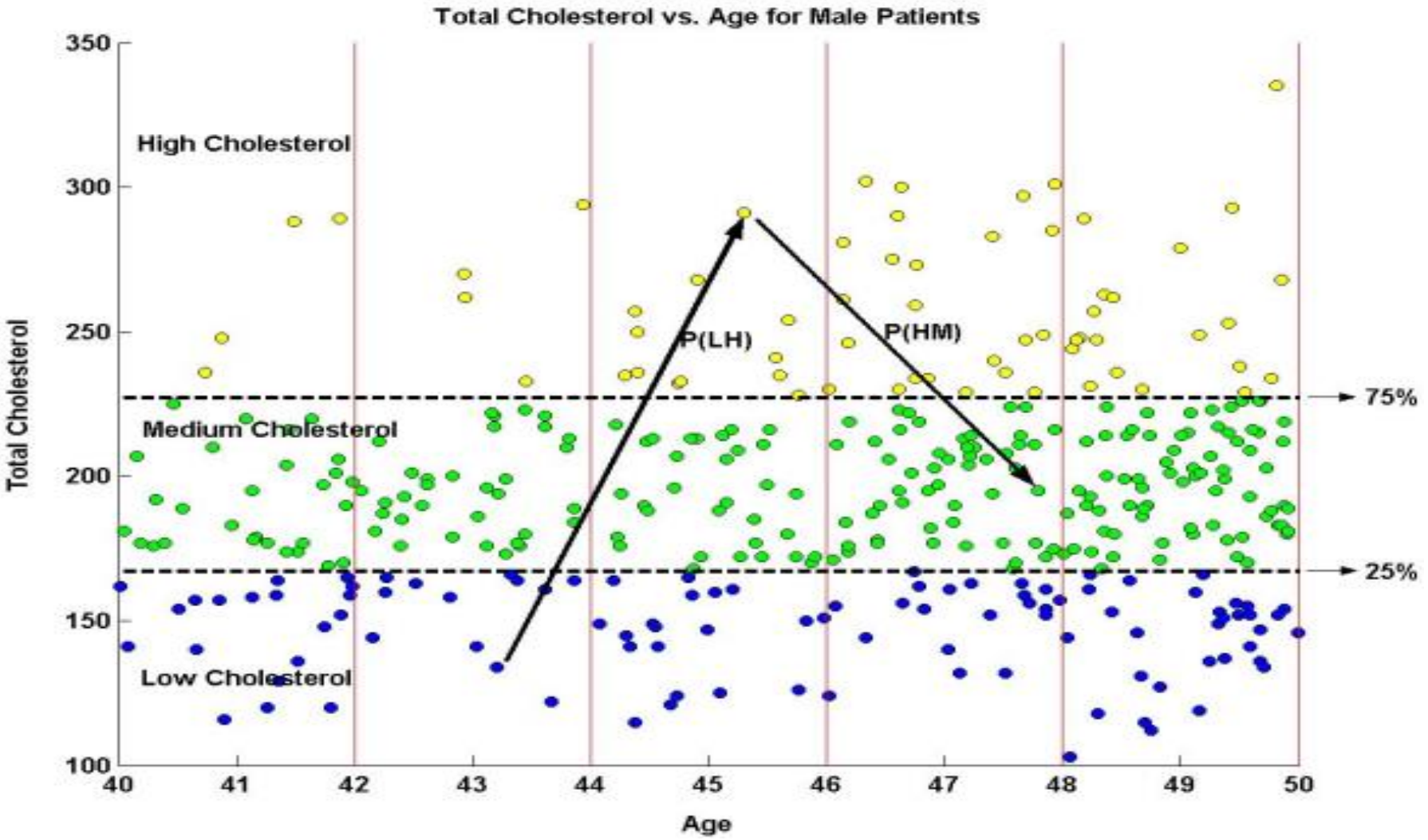
Transition probabilities between metabolic states:

- Electronic medical record data for total cholesterol (bad cholesterol) and HDL (good cholesterol) levels for many patients

Transition probabilities from healthy states to complication state

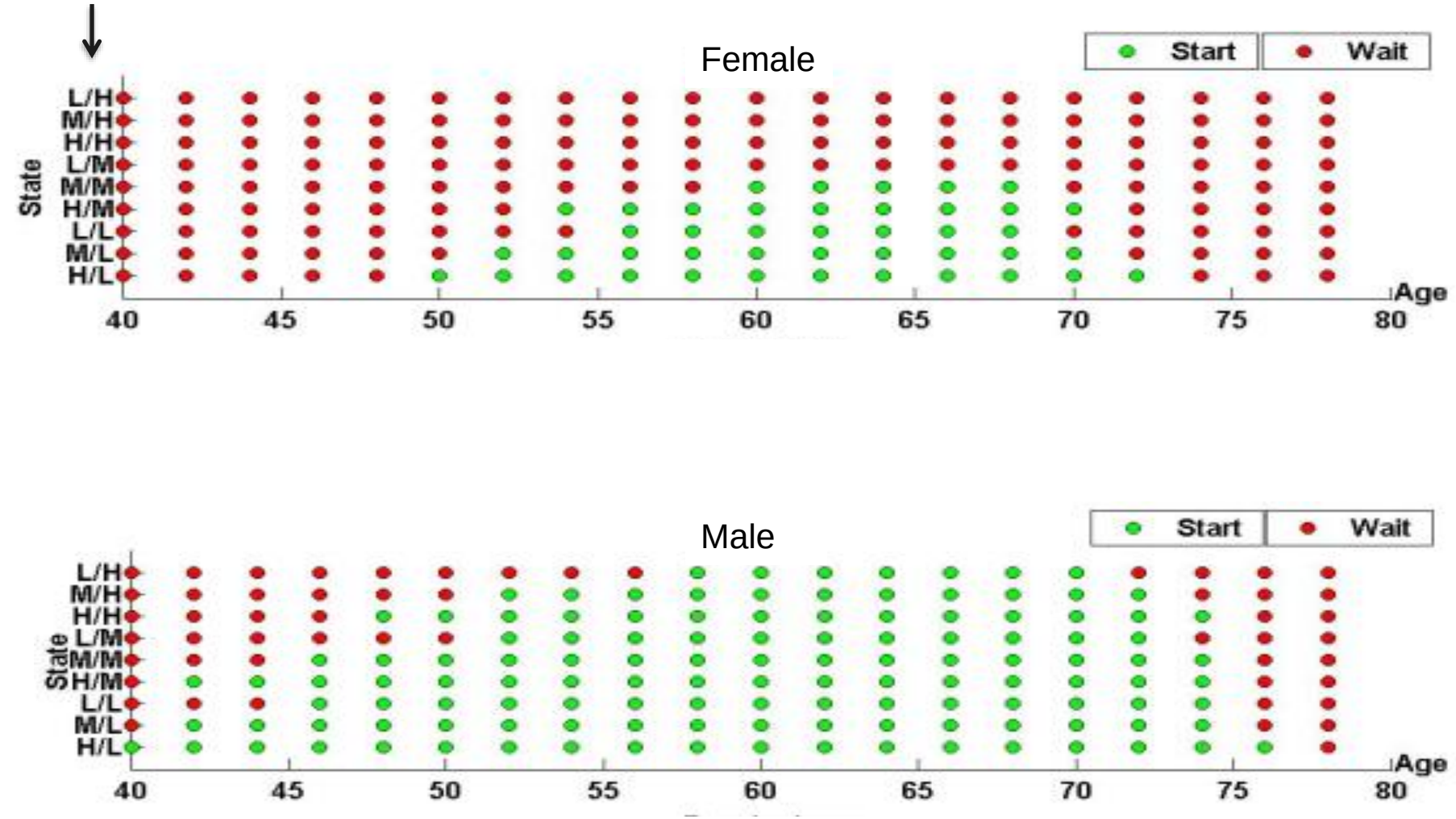
- Published cardiovascular risk models that estimate the probability of heart attack of stroke in the next year

Computing Transition Probabilities



Optimal Treatment Initiation Policy

Bad cholesterol/Good cholesterol



Other Related Examples

Example is based on this paper:

- Denton, B.T., **Kurt, M.**, Shah, N.D., Bryant, S.C., Smith, S.A., "A Markov Decision Process for Optimizing the Start Time of Statin Therapy for Patients with Diabetes," *Medical Decision Making*, 29(3), 351-367, 2008

Following are extensions of this work:

- **Kurt, M.**, Denton, B.T., Schaefer, A., Shah, N., Smith, S., "The Structure of Optimal Statin Initiation Policies for Patients with Type 2 Diabetes", *IIE Transactions on Healthcare* 1, 49-65, 2011
- **Mason, J.E.**, England, D., Denton, B.T., Smith, S., Kurt, M., Shah, N., "Optimizing Statin Treatment Decisions in the Presence of Uncertain Future Adherence," *Medical Decision Making* 32(1), 154-166, 2012.
- **Mason, J.**, Denton, B.T., Shah, N., Smith, S., "Optimizing the Simultaneous Management of Cholesterol and Blood Pressure Treatment Guidelines for Patients with Type 2 Diabetes," *European Journal of Operational Research*, 233, 727-738, 2013.

Sections

- Finite Horizon MDPs 
- Infinite Horizon MDPs
- Special Topics
  - Partially observable MDPs
  - Robust MDPs
  - Approximate DP
  - Reinforcement Learning

Infinite Horizon MDPs

Infinite horizon dynamic programs are unbounded decision process

We will discuss the following aspects of infinite horizon MDPs:

- Conditions under which they are well defined
- Optimality equations
- Solution methods
- Example

Why study infinite horizon MDPs?

- Sometimes they are a more accurate representation of a problem
- Sometimes they are **easier to solve** than finite horizon MDPs
- In some cases the optimal policy can be deduced without solving the MDP

Assumptions

We will assume the following when discussing infinite horizon MDPs:

- The set of actions, A, and states, S, are finite
- The decision maker's goal can be represented by additive rewards
- All problem data (transition probability matrix, rewards, state and action sets) are **stationary**

Total Expected Discounted Rewards

Given some policy π and initial state s the total expected discounted reward for a finite horizon MDP is:

$$v^{\pi}(s) = \mathrm{E}_{s}^{\pi}[\sum_{t=1}^{N} \lambda^{t-1} r_t(s_t, \pi(s_t))]$$

For an infinite horizon MDP:

$$v^{\pi}(s) = Lim_{N\to\infty} E_{s}^{\pi}[\sum_{t=1}^{N} \lambda^{t-1} r_t(s_t, \pi(s_t))]$$

where $0 < \lambda < 1$ is **required** for the limit to exist.

Notation

Given some decision rule $d(s)$ that is applied over an infinite horizon, let $r_d(s) \equiv r(s, d(s))$ and $\mathrm{p_d(j|s)} \equiv p(j|s, d(s))$

Let $\mathrm{r_d}$ denote the $|S|$ vector with $\mathrm{s^{th}}$ component $\mathrm{r_d(s)}$, referred to as the **reward vector**

Let $\mathrm{P_d}$ be the $|\mathrm{S}| \times |S|$ **transition probability matrix** with $(\mathrm{s}, j)^{\mathrm{th}}$ entry $\mathrm{p_d(j|s)}$.

The vector of expected values of the policy for each state is:

$$v^{\pi} = \sum_{t=1}^{\infty} \lambda^{t-1} P_d^{t-1} r_d$$

Policy Evaluation

Assuming a stationary policy $\pi = (d, d, \ldots)$, the expected value for each state can be expressed as the following **vector**:

$$v^{\pi} = \sum_{t=1}^{\infty} \lambda^{t-1} P_d^{t-1} r_d$$

$$= r_d + \lambda P_d r_d + \lambda^2 P_d P_d r_d + \cdots$$

$$= r_d + \lambda P_d \left(r_d + \lambda P_d r_d + \lambda^2 P_d^2 r_d + \cdots \right)$$

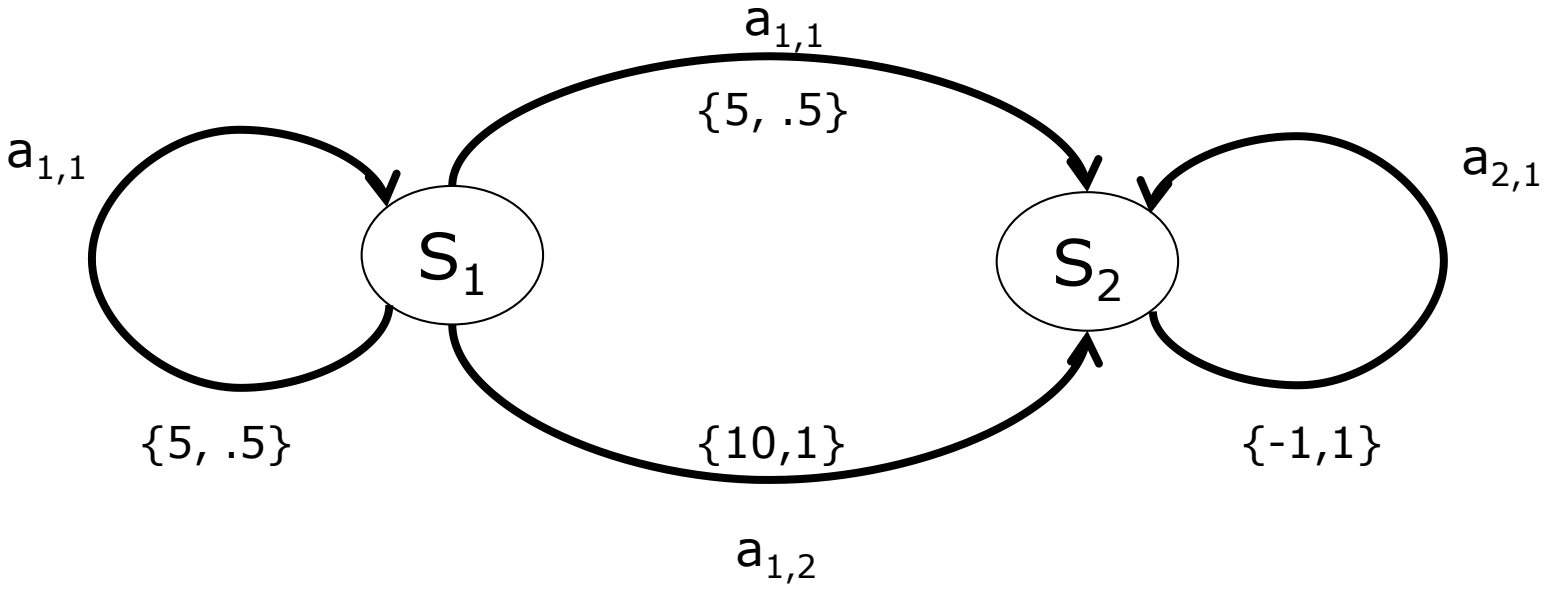
$$= r_d + \lambda P_d v^{\pi}$$

Thus the expected value of a policy is

$$v^{\pi} = (I - \lambda P_d)^{-1} r_d$$

Example: 2 State MDP

In state S_1 actions $a_{1,1}$ and $a_{1,2}$ are available; in state S_2 only $a_{2,1}$ is available. Rewards and transition probabilities are defined below as {r,p} At each stage the associated reward is received and then the transition occurs.



Exercise: Find $u_\lambda^\pi(s)$ for policies (a) $d^a(s_1) = a_{1,2}$ and $d^a(s_2) = a_{2,1}$ and (b) $\mathrm{d}^\mathrm{b}(s_1) = a_{1,1}$ and $d^b(s_2) = a_{2,1}$ using: $v^\pi = (I - \lambda P_d)^{-1} r_d$

Optimality Equations

The optimality equations for a stationary **finite horizon** MDP are

$$v_t(s) = \max_{a \in A} \{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v_{t+1}(j) \}, t = 1, \ldots, T-1$$

Boundary condition: $v_T(s) = r(s)$.

Passing to the limit as $t \to \infty$ for **infinite horizon** MDPs we have

$$v(s) = \max_{a \in A} \{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \}$$

Optimality Equations

The following optimality equations

$$v(s) = \max_{a \in A} \{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \}, \forall s$$

can be expressed in **vector notation** as $v = Lv$ where

$$\mathrm{L}v = \max_{d \in D} \{ r_d + \lambda P_d v \}$$

Note: L is a vector operator denoting the $\max_{d \in D} \{\cdot\}$ operation

Fundamental Result

Theorem : The solution to the equations:

$$v = Lv$$

yields the maximum expected discounted rewards and the optimal policy.

Importance: This provides a means to find the optimal value function and thus the optimal policy for an infinite horizon MDP.

Question: How can we solve these equations?

Solution to Optimality Equations

Theorem: If L is a contraction mapping, then for arbitrary v^0 the sequence $\{\mathrm{v}^0, \ldots, \mathrm{v}^n\}$ defined by $\mathrm{v}^{n+1} = \mathrm{L}\mathrm{v}^n$ converges to v^* as $\mathrm{n} \to \infty$, where $v^* = Lv^*$.

Theorem: Let $\{\mathrm{v}^n\}$ satisfy $\mathrm{v}^{n+1} = \mathrm{L}\mathrm{v}^n$, then:

$$||\mathrm{v}^{n+1} - \mathrm{v}^*|| \leq \epsilon/2 \text{ whenever} ||\mathrm{v}^{n+1} - v^n|| < \frac{\epsilon(1-\lambda)}{2\lambda}$$

Importance: these theorems gives rise to the well know **value iteration algorithm**.

Value Iteration

Algorithm (Value Iteration) :

1. Select some intial vector v^0 *and set* $n = 0$.

2. *Apply* $v^{n+1} = Lv^n$

3. *If* $||v^{n+1} - v^n|| < \epsilon(1 - \lambda)/2\lambda$ *go to step 4. Otherwise* $n=n+1$ *and return to step 2.*

4. $d_\epsilon(s) \in argmax_{a \in A} \{\mathrm{r(s,a)} + \sum_{j \in S} p(j|s,a)v^n(j)\}, for\ all\ s.$

Value Iteration – Why Does it Work?

Definition: Operator L is a **contraction mapping** if there exists a $0 < \lambda < 1$ such that

$$||Lv - Lu|| \leq \lambda ||v - u||$$

for all u and v.

Basic Idea: Applying L to two vectors brings them closer together.

Proposition: If $0 < \lambda < 1$ then L is a contraction mapping.

Observations About Value Iteration

- Convergence of the optimal value function is influenced by the discount factor
  - The higher the discount factor the slower the rate of convergence
- In some cases the optimal policy may be found very quickly but stopping criteria may require a large number of iterations
- Advanced approaches seek to reduce the number of iterations necessary to find an ϵ-optimal policy

Policy Iteration

Policy iteration is an alternative to value iteration for solving infinite horizon MDPs

Basic Idea:

Step 1: *Choose an initial policy*

Step 2: *Evaluate the policy*

Step 3: *Find a better policy*

Step 4: *If the new policy is optimal then stop. Otherwise go to step 2.*

Policy Iteration

Algorithm (Policy Iteration):

1. Set $n = 0$ and select an arbitrary decision rule $d_0 \in D$.

2. ***Policy Evaluation***: $Obtain\ v^n\ by\ solving\ (I - \lambda P_{d_n})v = r_{d_n}$

3. ***Policy Improvement***: $Choose\ d_{n+1}\ to\ satisfy$:

$$d_{n+1} \in argmax_{d \in D}\{r_d + \lambda P_d v^n\},$$

setting $d_{n+1} = d_n$ if possible.

4. If $d_{n+1} = d_n$, stop and set $d^* = d_n$. Otherwise $n = n + 1$ and return to step 2.

Policy Iteration – Why does it work?

Proposition: Let v^{n} and $\mathrm{v}^{\mathrm{n+1}}$ be successive values generated by the policy iteration algorithm. Then $\mathrm{v}^{\mathrm{n+1}} \geq v^{n}$.

Importance: This means **policy iteration** generates a finite sequence of improving decision rules $\{\mathrm{d}_{\mathrm{n}}\}$, and value functions $\{\mathrm{v}_{\mathrm{n}}\}$.

Policy Iteration – Why Does it Work?

Theorem: The policy iteration algorithm terminates in a finite number of iterations, with a solution of the optimality equations and an optimal policy, π^*.

Observations:

- If an improvement to a policy is possible then policy iteration will find it
- If no improvement is possible then the algorithm terminates with a (optimal) solution to $v = Lv$

Value Iteration vs Policy Iteration

- Value iteration generates ϵ- optimal solutions; Policy iteration is exact.

- Policy iteration solves a system of linear equations at each iteration which can be computationally prohibitive for very large problems

- There are many variants of value and policy iteration that can accelerate convergence. The most well known is **modified policy iteration**.

Linear Programming Method

MDPs can be formulated as linear programs:

- Decision variables: value function for each state ($v(s)$)
- Constraints: optimality equations

Advantages of this approach are:

1. Insights can be gained by viewing MDPs through the “lens” of linear programming
2. Advances in large-scale linear programming offer advantages to solving MDPs efficiently
3. Useful for formulating **constrained MDPs**

Linear Programming Method

The following linear program (LP) formulation determines an optimal policy for a maximization problem

$$\text{Min } \sum_{j \in S} \alpha(j) v(j)$$

$$s.t.$$

$$v(s) - \sum_{j \in S} \lambda p(j|s,a) v(j) \geq r(s,a), \ \forall a \in A(s), s \in S$$

$$v(s) \ \ urs, \forall s$$

where $\alpha(j), j \in S$ are positive scalars, often chosen to represent the initial state probability distribution ($\sum_{j \in S} \alpha(j) = 1$)

Infinite Horizon MDP Example

MANAGEMENT SCIENCE
Vol. 50, No. 10, October 2004, pp. 1420–1430
ISSN 0025-1909 | EISSN 1526-5501 | 04 | 5010 | 1420

informs

DOI 10.1287/mnsc.1040.0287

The Optimal Timing of Living-Donor Liver Transplantation

Oguzhan Alagoz
Department of Industrial Engineering, University of Pittsburgh, Pittsburgh, Pennsylvania 15261, alagoz@ie.pitt.edu

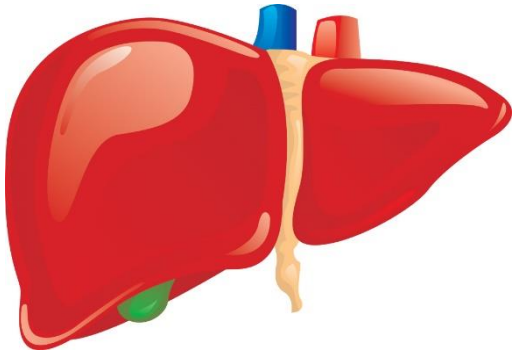
Lisa M. Maillart
Weatherhead School of Management, Case Western Reserve University, Cleveland, Ohio 44106, lisa.maillart@case.edu

Andrew J. Schaefer
Departments of Industrial Engineering and Medicine, University of Pittsburgh,
Pittsburgh, Pennsylvania 15261, schaefer@ie.pitt.edu

Mark S. Roberts
Division of General Internal Medicine, University of Pittsburgh, Pittsburgh, Pennsylvania 15213, robertsm@upmc.edu

Living Donor Liver Transplants

Living donors are a new and increasing source of organs for liver transplantation

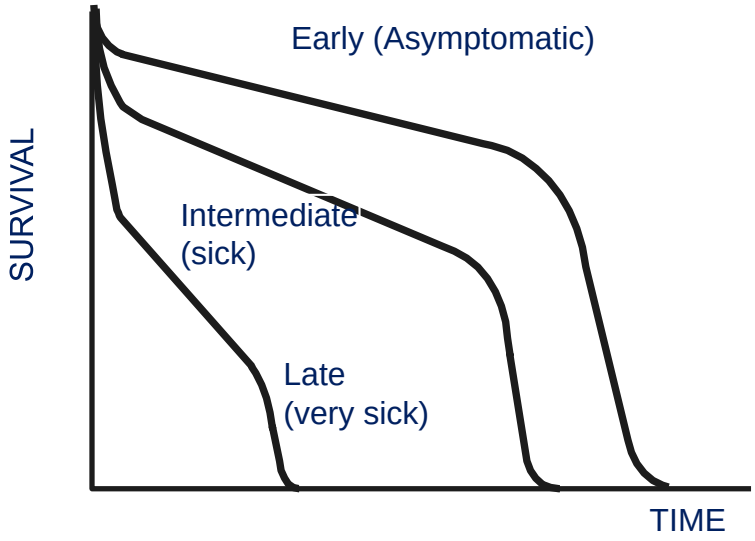


Using living donors has several advantages:

- Increases the supply of organs
- Quality of the organ and transplant tends to be higher
- Cold ischemia time is reduced
- Transplantation occurs at the discretion of the doctor

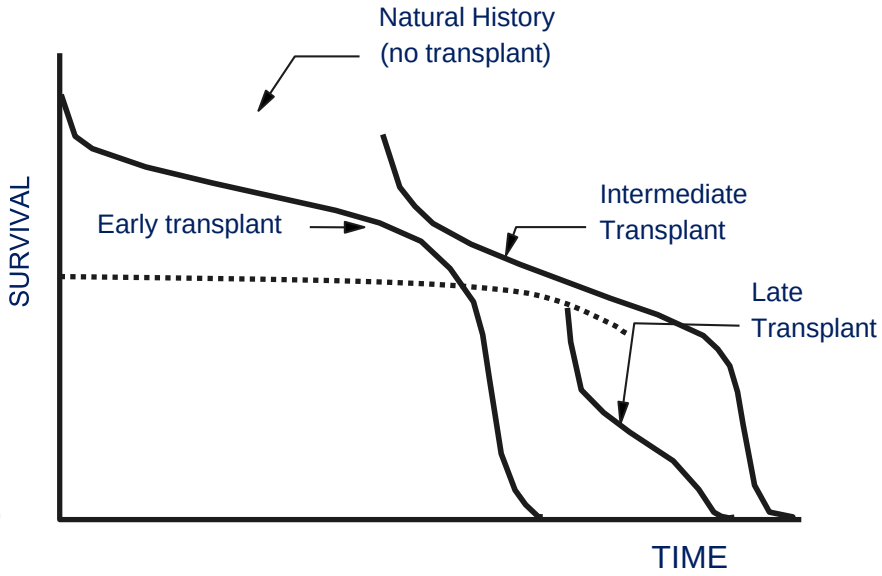
What is the Optimal Time to Transplant?

Immediate Transplant



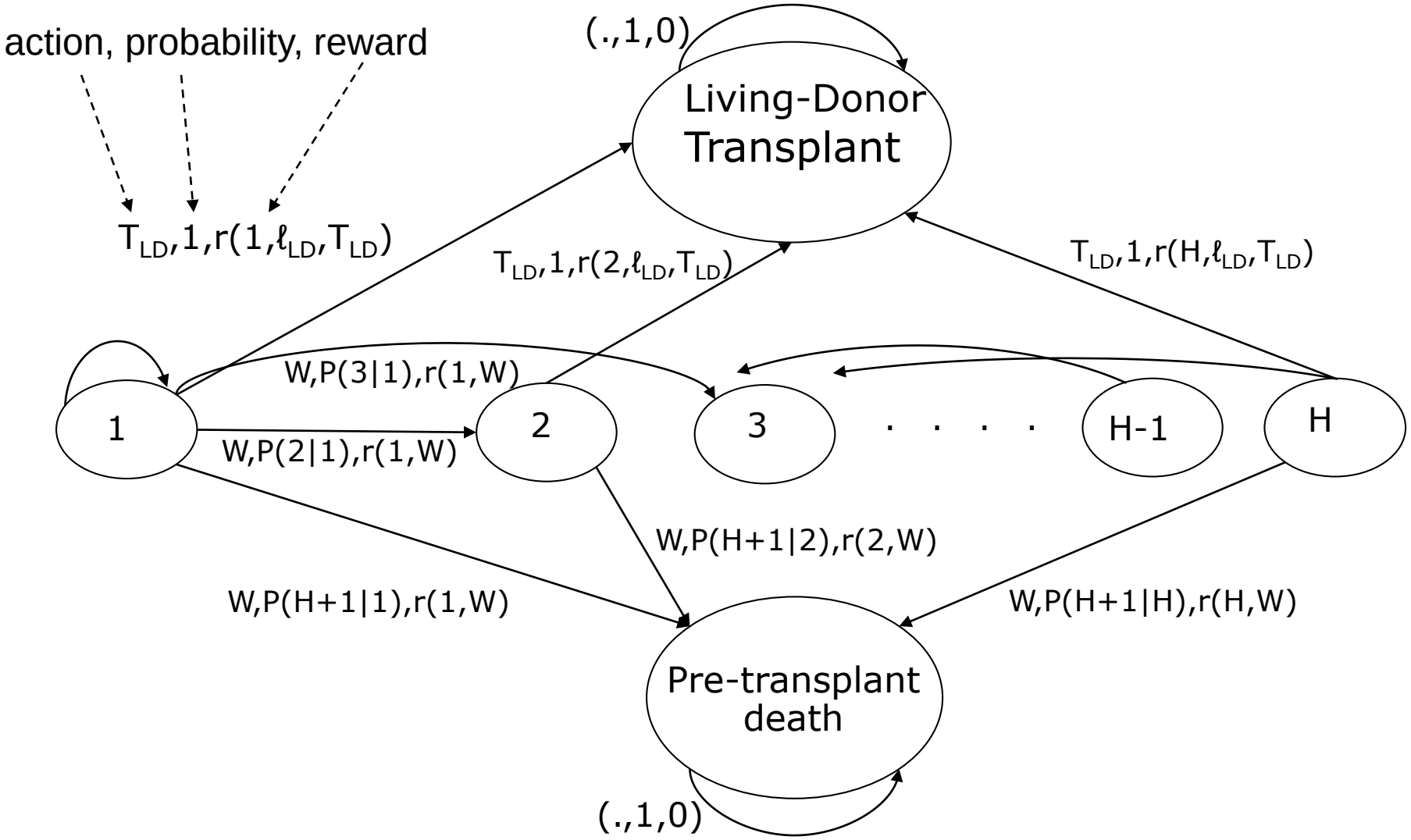
Transplantation Survival
by stage of disease

Delayed Transplant



Transplantation Survival vs. Natural History

Markov Chain



Optimality Equations

Transplant

$$\underbrace{V(h)}_{\text{Max survival under optimal policy}} = Max \left\{ \begin{array}{l} r(h, \ell_{LD}, T_{LD}), \\ \underbrace{r(h, W) + \lambda \sum_{h'} P(h' \mid h) V(h')}_{\text{Defer Transplant}} \end{array} \right\}$$

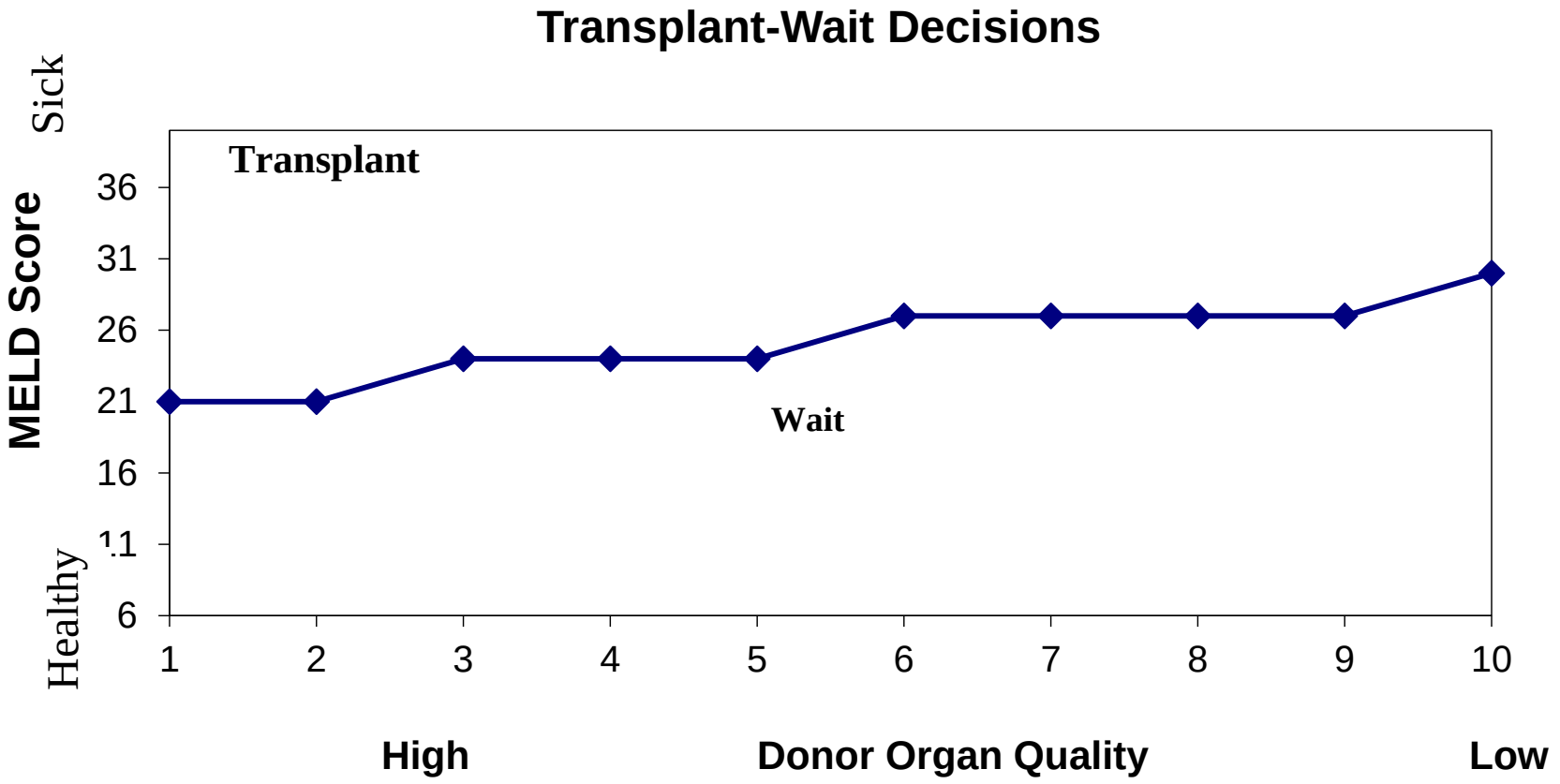
Properties:

- Sufficient conditions for immediate transplant
- Sufficient conditions for optimality of a control-limit policy

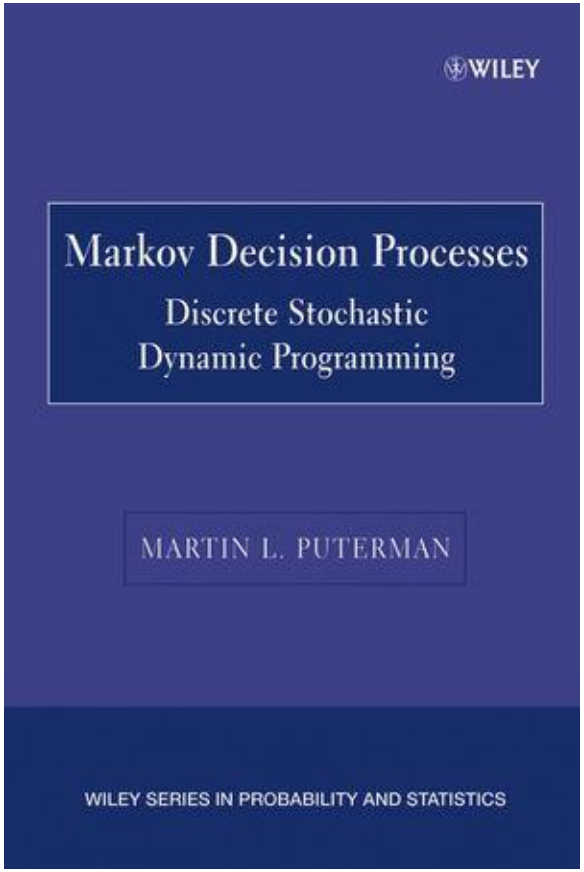
Data Sources

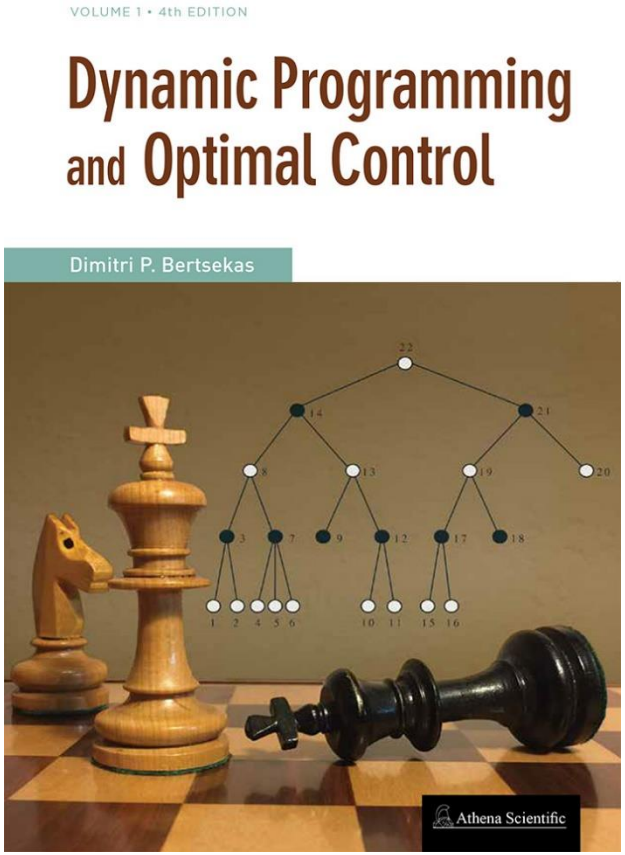
1. University of Pittsburgh Medical Center - large transplant center with detailed clinical data for >3000 patients including MELD score.
    - Estimate of Transition Probabilities
2. UNOS – nationwide data for >28,000 patients including transplant survival rates
    - Rewards

Optimal Policy



MDPs: Where to learn more





Sections

- Finite Horizon MDPs 
- Infinite Horizon MDPs 
- Special Topics
  - Partially observable MDPs
  - Robust MDPs
  - Approximate DP
  - Reinforcement Learning

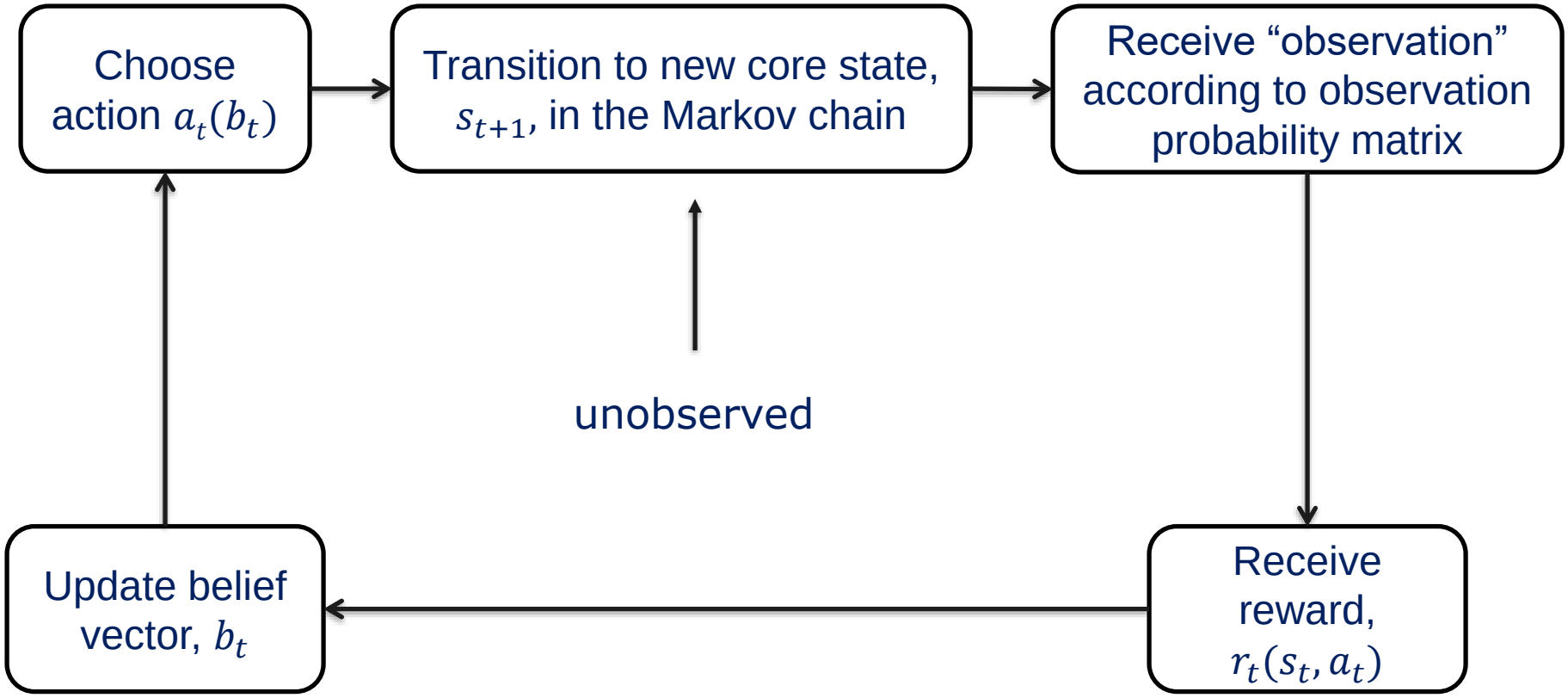
Partially Observable MDPs (POMDPs)

Model Elements:

- Decision Epochs: $t = 1, \ldots, T$
- Core States: $s_t \in S$
- Actions: $a_t \in A$
- Rewards: $r_t(s_t, a_t)$
- Transition Probability Matrix: P
- Observations: $o \in O$
- Observation Probability Matrix: $Q \in R^{|S| \times |O|}$

} Unique to POMDPs

POMDP Sequence of Events



Sufficient Statistic

The belief vector is a vector with one element for each state that defines the probability the system is in state s_t

$$b_t(s_t) = P(s_t | \underbrace{o_t, a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_1, a_0}_{h_t})$$

Complete history of observations up to and including t as well as history of actions up to and including time t-1

For POMDPs, the *belief vector*, b, is a "sufficient statistic" to define the optimal policy.

Some Basic Properties

Bayes Theorem:

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

Conditional Probability:

$$\mathrm{P}(A,B) = P(A|B)P(B)$$

Law of total probability:

$$P(A) = \sum_{i=1}^{n} P(A, B_i)$$

If events $B_1, B_2, \ldots, B_n$ are mutually exclusive.

Bayesian Updating

Belief Update Formula:

$$b_t(s_t) = P(s_t | o_t, a_{t-1}, o_{t-1}, a_{t-2}, \dots, o_1, a_0)$$

$$= \frac{P(s_t, o_t, a_{t-1} |\, o_{t-1}, a_{t-2}, \dots, o_1, a_0)}{P(o_t, a_{t-1} |\, o_{t-1}, a_{t-2}, \dots, o_1, a_0)}$$

Numerator:

$$P(s_t, o_t, a_{t-1} |\, \overbrace{o_{t-1}, a_{t-2}, \dots, o_0, a_0}^{h_{t-1}}) = \sum_{s_{t-1} \in S} P(s_t, o_t, a_{t-1}, s_{t-1} |\, h_{t-1})$$

$$= \sum_{s_{t-1} \in S} P(o_t | s_t, a_{t-1}, s_{t-1}, h_{t-1})\, P(s_t | a_{t-1}, s_{t-1}, h_{t-1}) P(a_{t-1} | s_{t-1}, h_{t-1}) P(s_{t-1} | h_{t-1})$$

$$= P(a_{t-1} | h_{t-1}) P(o_t | s_t) \sum_{s_{t-1} \in S} P(s_t | a_{t-1}, s_{t-1})\, b_{t-1}(s_{t-1})$$

Bayesian Updating

Belief Update Formula:

$$b_t(s_t) = \frac{P(s_t, o_t, a_{t-1} |\, o_{t-2}, a_{t-2}, \dots, o_0, a_0)}{P(o_t, a_{t-1} |\, o_{t-2}, a_{t-2}, \dots, o_0, a_0)}$$

Denominator:

$$P(o_t, a_{t-1} |\, \overbrace{o_{t-1}, a_{t-2}, \dots, o_0, a_0}^{h_{t-1}}) = \sum_{s_t' \in S} \sum_{s_{t-1} \in S} P(s_t', o_t, a_{t-1}, s_{t-1} |\, h_{t-1})$$

$$= \sum_{s_t' \in S} \sum_{s_{t-1} \in S} P(o_t | s_t', a_{t-1}, s_{t-1}, h_{t-1})\, P(s_t' | a_{t-1}, s_{t-1}, h_{t-1}) P(a_{t-1} | s_{t-1}, h_{t-1}) P(s_{t-1} | h_{t-1})$$

$$= P(a_{t-1} | h_{t-1}) \sum_{s_t' \in S} P(o_t | s_t') \sum_{s_{t-1} \in S} P(s_t' | a_{t-1}, s_{t-1})\, b_{t-1}(s_{t-1})$$

Bayesian Updating

.

Now everything is in terms of transition probabilities, observation probabilities, and the prior belief vector

$$b_t(s_t) = \frac{P(o_t|s_t)\sum_{s_{t-1}\in S} P(s_t|s_{t-1},a_{t-1})\, b_{t-1}(s_{t-1})}{\sum_{s_t'\in S} P(o_t|s_t')\sum_{s_{t-1}\in S} P(s_t'|s_{t-1},a_{t-1})\, b_{t-1}(s_{t-1})}$$

Numerator: Probability of observing o_t and system is in s_t

Denominator: Probability of observing o_t

Optimality Equations for POMDPs

Rewards Vector: $\mathrm{r}_t(a_t) = (r_t^1(a_t), \ldots, r_t^S(a_t))'$ denotes the expected rewards under transitions and observations

$$r_t^{s_t}(a_t) = \sum_{o_{t+1} \in O} \sum_{s_{t+1} \in S} r(s_t, a_t, s_{t+1}, o_{t+1}) p(s_{t+1}|s_t, a_t) p(o_{t+1}|s_{t+1})$$

Optimality Equations: In POMDPs, the value function is defined on the belief space.

Probability of observation o_{t+1} given belief vector b_t and action a_t

$$v_t(b_t) = \max_{a_t \in \mathcal{A}} \left\{ b_t \cdot r_t(a_t) + \lambda \sum_{o_{t+1} \in \mathcal{O}} \gamma(o_{t+1}|b_t, a_t) v_{t+1}(T(b_t, a_t, o_{t+1})) \right\}$$

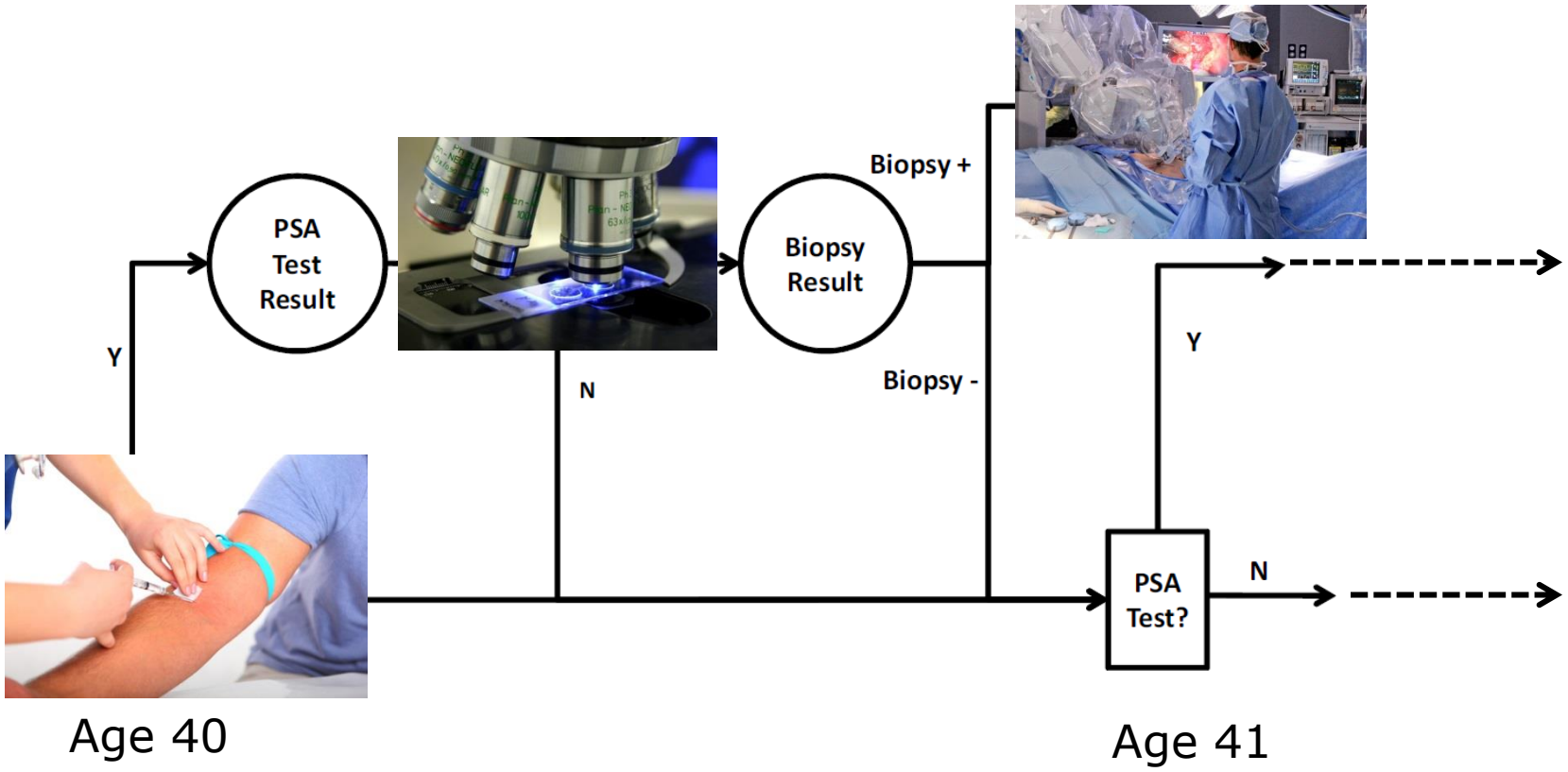
Boundary Condition: $v_{T+1}(b_{T+1}) = b_{T+1} \cdot r_{T+1}$

Updated belief given observation o_{t+1} and action a_t

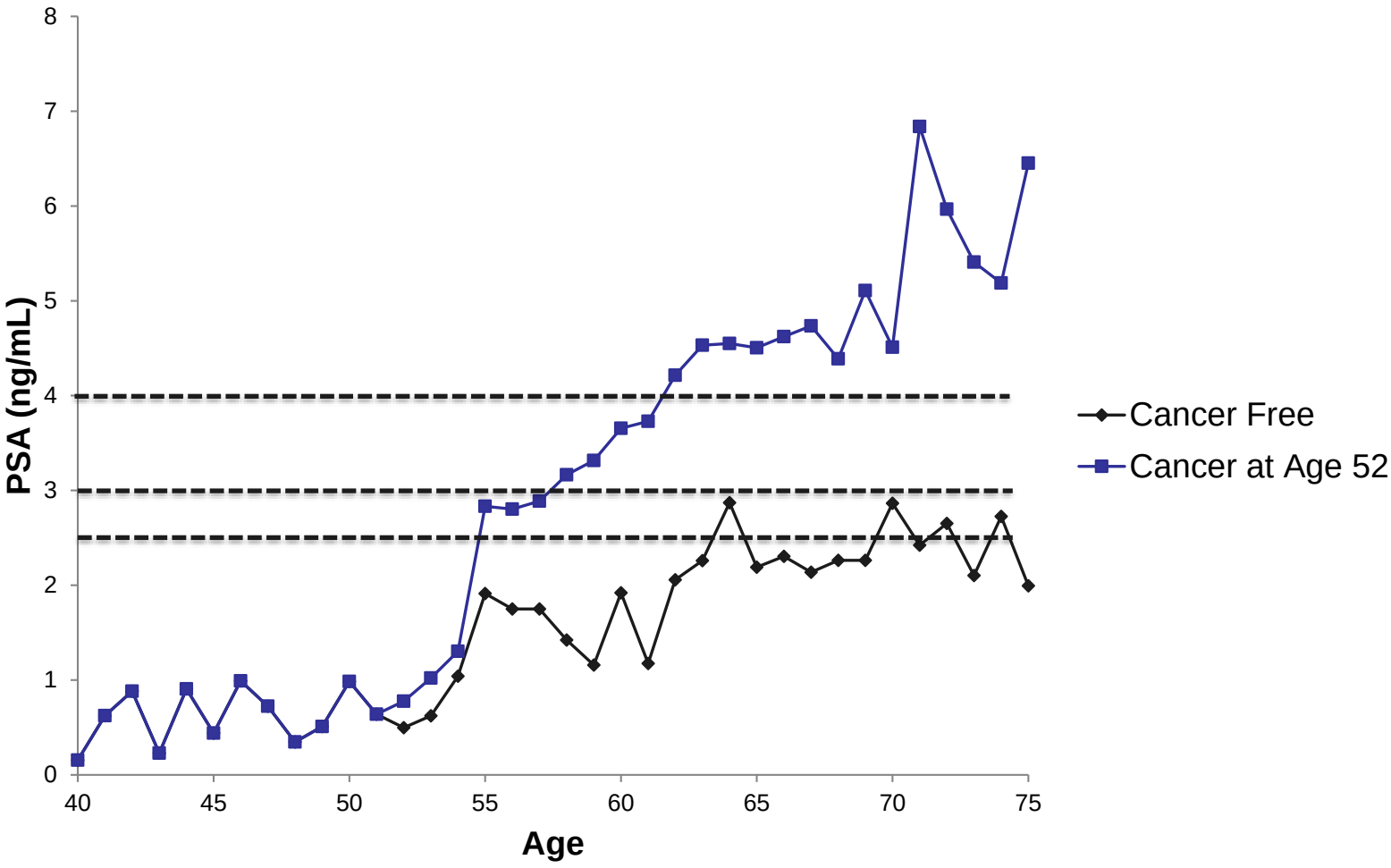
Solution Methods

- POMDPs are difficult to solve exactly:
  - Time complexity is exponential in the number of actions, observations, and decision epochs
  - Dimensionality in the state space grows with the number of core states
- Complexity class is **P-Space Hard**
- Most approaches rely on approximations: finite grids, supporting hyperplane sampling.

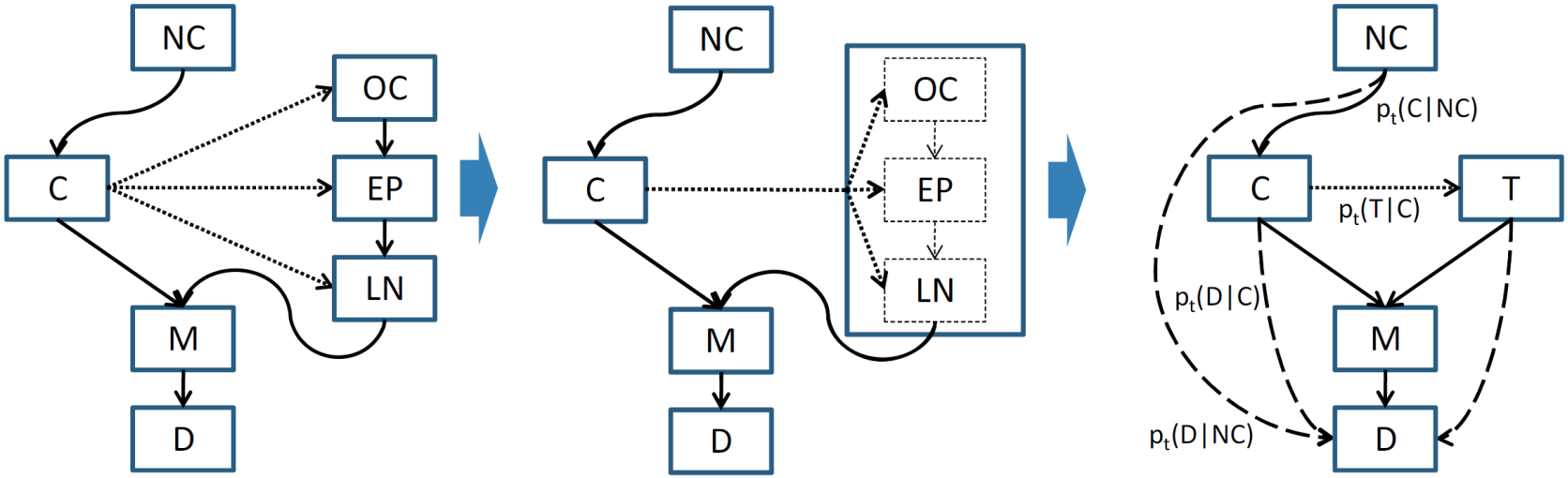
POMDP Example: Prostate Cancer



Biomarker Test: PSA



Core States



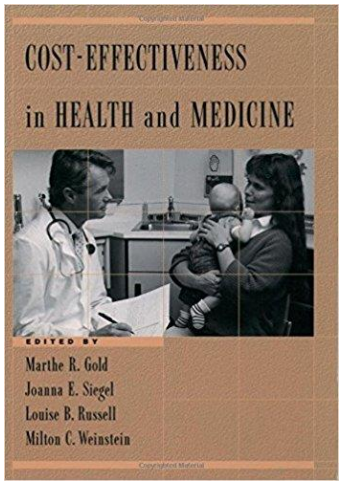
Markov transitions between prostate cancer states:

- No cancer (NC) → Unobservable
- Cancer present but not detected (C) → Unobservable
- Cancer detected (T) → Treated immediately after detected
- Death (D) → Prostate cancer and other cause mortality

Model Description

- Decision Epochs, $t = 40\,41, \ldots, 85$
- Health States: Health/cancer status, s_t
- Observations: PSA test result, o_t
- Observation Matrix: $q_t(o_t|s_t)$
- Rewards: Quality adjusted life years
  - $r_t(NC, No\ PSA\ Test) = 1$
  - $r_t(NC, PSA\ Test) = 1 - \delta$
  - $r_t(NC, Biopsy) = 1 - \mu$
  - $r_t(C, No\ PSA\ Test) = 1$
  - $r_t(C, PSA\ Test) = 1 - \delta$
  - $r_t(C\ Biopsy) = 1 - \mu - f\epsilon$

Resource to learn more about QALYs and other public health measures:



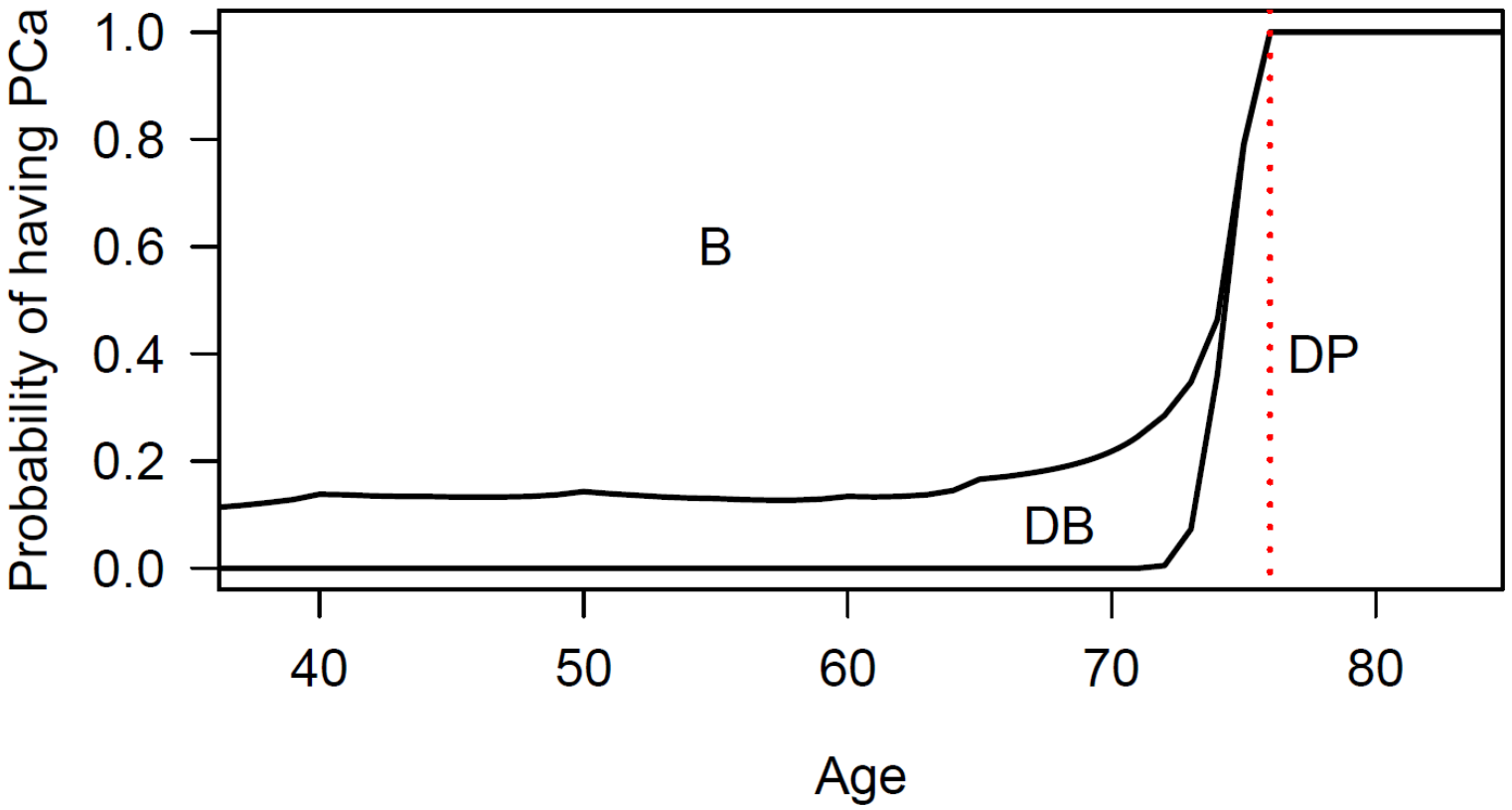
Model Data

11,872 patients from Olmsted county, MN with age, PSA, biopsy and cancer information from 1993 through 2006

Population size	11,872
Age: Mean(SD)	63.0(12.7)
Race	
Caucasian	96%
Other	4%
Outcomes	
Prostate biopsy	908
Prostate cancer diagnosis	628

Other parameters are drawn from the medical literature

Optimal Policy for Screening



Zhang, J., Denton, B.T. Balasubramanian, H., Shah, N.D., and Inman, B.A.. 2012. "Optimization of prostate biopsy referral decisions." *M&SOM,* 14(4); 529-547.

POMDPs: Where to learn more

- Tutorial: “POMDPs for Dummies” http://cs.brown.edu/research/ai/pomdp/tutorial/

- Smallwood, Richard D., and Edward J. Sondik. "The optimal control of partially observable Markov processes over a finite horizon." *Operations research* 21, no. 5 (1973): 1071-1088.

- Sondik, Edward J. "The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs." *Operations research* 26, no. 2 (1978): 282-304.

- Monahan, George E. "State of the art—a survey of partially observable Markov decision processes: theory, models, and algorithms." *Management Science* 28, no. 1 (1982): 1-16.

- Kaelbling, Leslie Pack, Michael L. Littman, and Anthony R. Cassandra. "Planning and acting in partially observable stochastic domains." *Artificial intelligence* 101, no. 1 (1998): 99-134.

Robust MDPs

- All models are subject to uncertainty in model parameter estimates and model assumptions
  - Transition probabilities are based on statistical estimates from longitudinal data
  - Rewards are based on estimates of mean patient utility, cost, or other performance measures
- Robust MDPs (RMDPs) attempt to account for this uncertainty

RMDP Formulation

- Goal of a standard finite horizon MDP is to find π^* with respect to a fixed Markov chain with TPM, P:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}}\, E^P\left[\sum_{t=1}^{N-1} r_t(s_t, \pi(s_t)) + r_N(s_N)\right]$$

- An RMDP assumes TPM is restricted to lie in an uncertainty set, U, leading to the following optimality equations:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \min_{P \in U} E^P\left[\sum_{t=1}^{N-1} r_t(s_t, \pi(s_t)) + r_N(s_N)\right]$$

Alternative Formulations

RMDPs can be viewed as a stochastic game against an "adversary." Following are two important variants.

Time Invariant Case – Adversary selects a single TPM

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \min_{P \in U} E^P[\sum_{t=1}^{N-1} r_t(s_t, \pi(s_t)) + r_N(s_N)]$$

Time Varying Case – Adversary selects a TPM at each epoch

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \min_{P_t \in U} E^{P_t}[\sum_{t=1}^{N-1} r_t(s_t, \pi(s_t)) + r_N(s_N)]$$

Time Varying Model

The time varying case can be viewed as a sequential game against an **adversary.** This problem is "easy" to solve when the **rectangularity assumption** is made:

$$U = \prod_{\forall s_t \in S} U(s_t)$$

Under this assumption the optimality equations are:

$$v_t(s_t) = \max_{a_t \in A} \left\{ r_t(s_t, a_t) + \max_{p(s_t) \in U(s_t)} \lambda \sum_{s_{t+1} \in S} p(s_{t+1}|s_t, a_t)\, v_{t+1}(s_{t+1}) \right\}$$

Where $p(s_t)$ is the row of the TPM corresponding to state s_t and $U(s_t)$ is the row's uncertainty set.

Uncertainty Sets

Many choices of U have been proposed:

- Finite scenario model:
- $$U(s_t) = \{p^1(s_t), p^2(s_t), \dots, p^K(s_t)\}$$
- Interval model:

$$U(s_t) = \{p(s_t) | \underline{p}(s_t) \leq p(s_t) \leq \overline{p}(s_t), p(s_t) \cdot \mathbf{1} = 1\}$$

- Ellipsoidal models, relative entry bounds, …

RMDP Case Study: Type 2 Diabetes

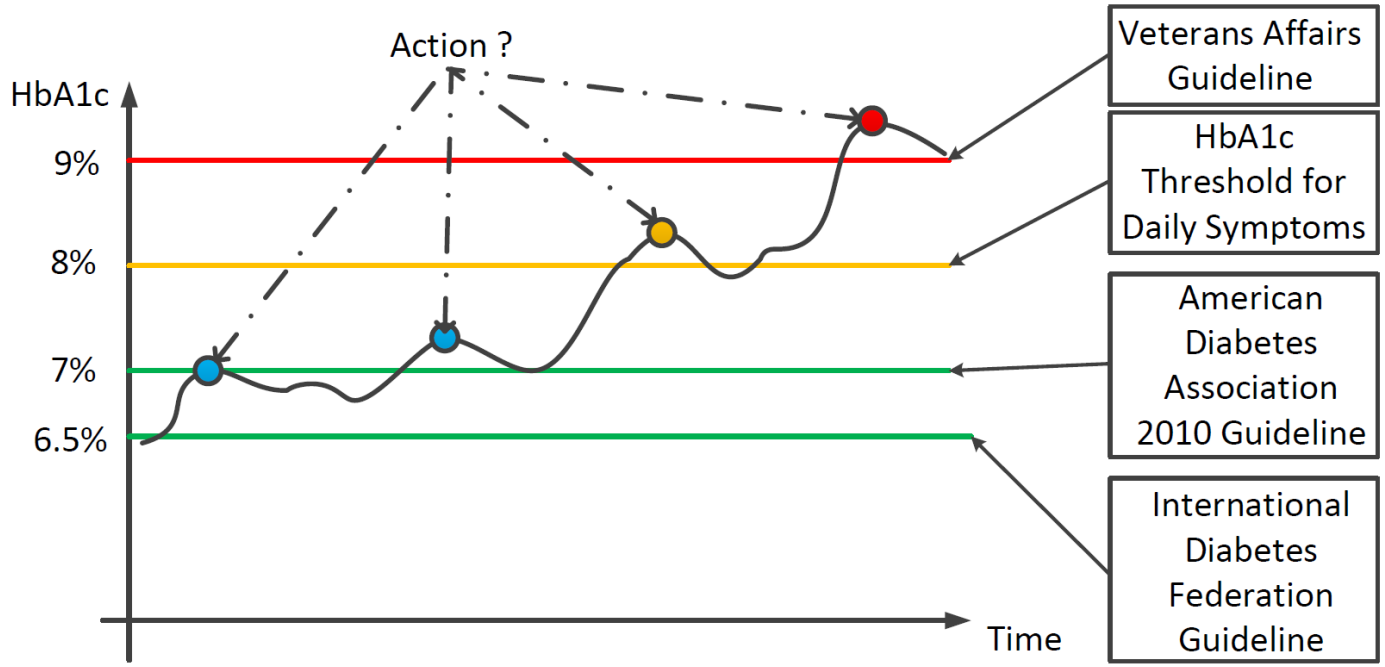
Many medications that vary in efficacy, side effects and cost.

Oral Medications:
- Metformin
- Sulfonylurea
- DPP-4 Inhibitors

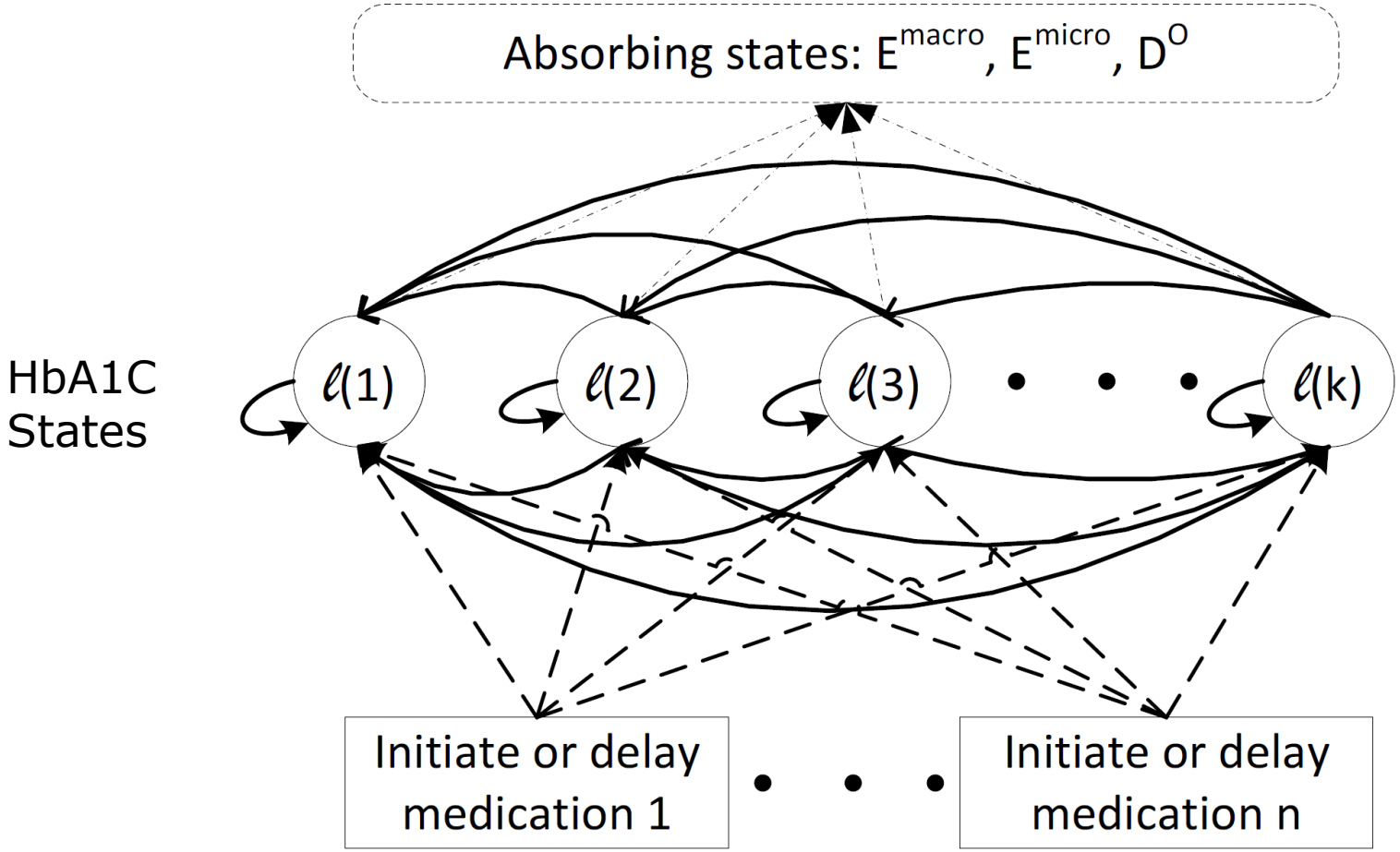
Injectable Medications:
- Insulin
- GLP-1 Agonists

Treatment Goals

- HbA1C is an important biomarker for blood sugar control
- But disagreement exists about the optimal goals of treatment and which medications to use



Markov Chain for Type 2 Diabetes



Estimating the Uncertainty Set

A combination of laboratory data and pharmacy claims data was to estimate transition probabilities between deciles

$$p(s'|s\ ,a) = \frac{n(s,s',a)}{\sum_{s'} n(s,s',a)}, \forall s', s, a$$

$1 - \alpha$ confidence intervals for row s of the TPM:

$$[\hat{p}(s'|s,a) - S(\hat{p}(s'|s,a)L, \quad \hat{p}(s'|s,a) + S(\hat{p}(s'|s,a)L]$$

where

$$S(\hat{p}(s'|s,a)L = \left[\chi^2_{|s|-1,\alpha/2|S|)} \frac{\hat{p}(s'|s,a)\big(1-\hat{p}(s'|s,a)\big)}{N(s)}\right]^{\frac{1}{2}}$$

Uncertainty Set with Budget

$$U(s_t) = \left\{ \begin{array}{l} p(s_{t+1}|s_t) = \hat{p}(s_{t+1}|s_t) - \delta^L z^L(s_{t+1}) + \delta^U z^U(s_{t+1}), \quad \forall s_{t+1} \\ \sum_{s_{t+1} \in S} p(s_{t+1}|s_t) = 1 \\ \boxed{\sum_{s_{t+1}} (z^L(s_{t+1}) + z^U(s_{t+1})) \leq \Gamma(s_{t+1})} \\ z^L(s_{t+1}) \cdot z^U(s_{t+1}) = 0, \quad \forall s_{t+1} \\ 0 \leq p(s_{t+1}|s_t) \leq 1, \quad \forall s_{t+1} \end{array} \right.$$

Properties:

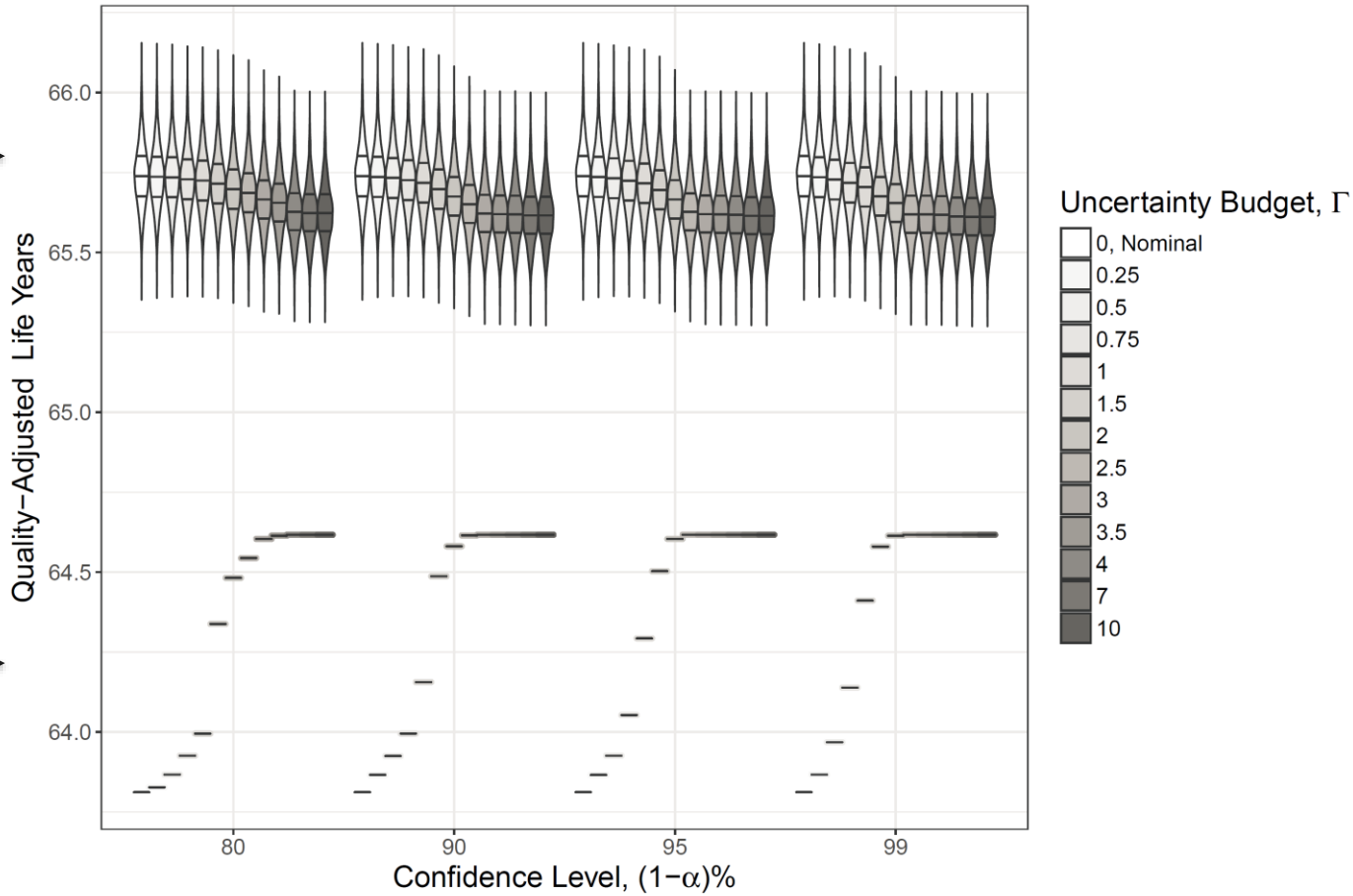
- Can be reformulated as a linear program
- For $\Gamma = |S|$ can be solved in $O(|S|)$

Results

Quality adjusted life years to first health complications for women with type 2 diabetes

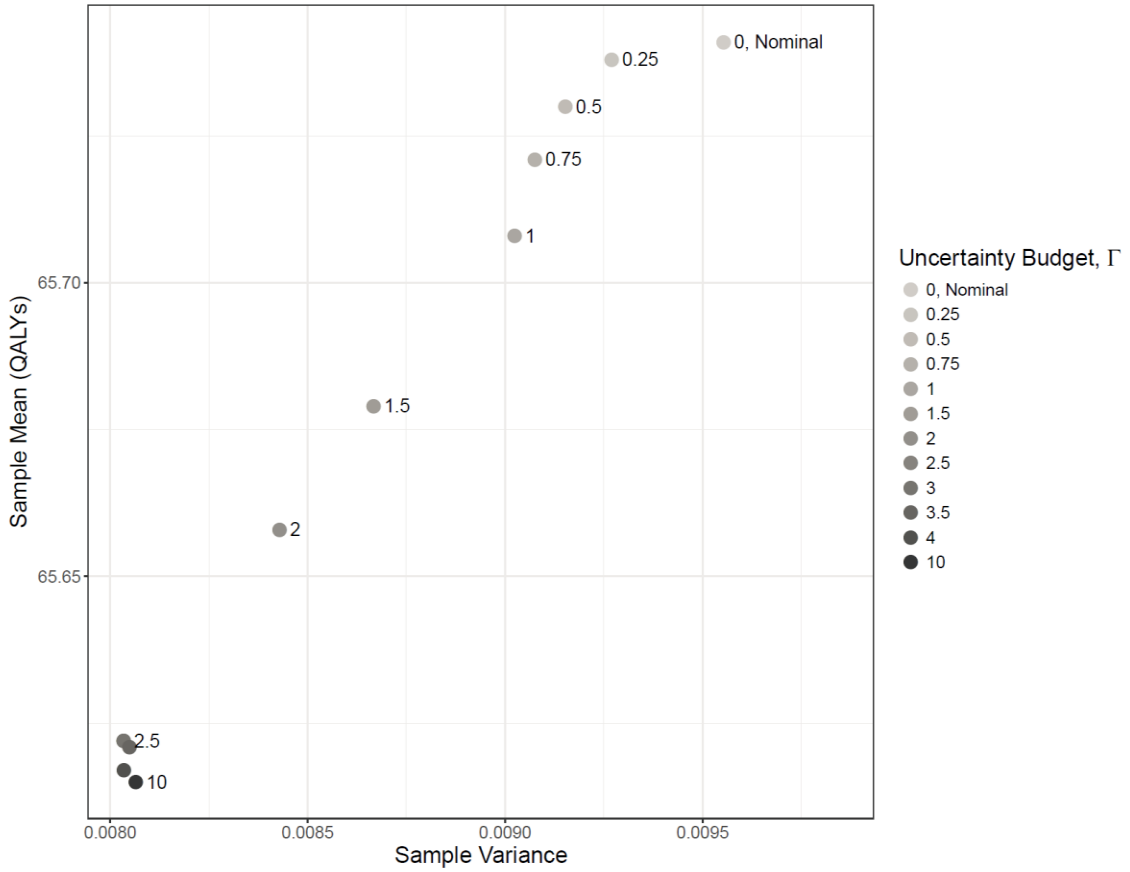
Simulated →

Worst-case →



Results

Mean QALYs versus variance in QALYs to first event for women with type 2 diabetes



RMDPs: Where to learn more

- Nilim, A., and El Ghaoui, L. 2005. "Robust control of Markov decision processes with uncertain transition matrices." *Operations Research* 53(5); 780-798.

- Iyengar, G.N. 2005. "Robust dynamic programming." *Mathematics of Operations Research* 30(2); 257-280.

- Wiesemann, W., Kuhn, D., and Rustem, B. 2013. "Robust Markov decision processes." *Mathematics of Operations Research* 38 (1); 153-183.

- Delage, E., Iancu, D. 2015. “Robust Multistage Decision Making.” INFORMS Tutorials in Operations Research

Approximate Dynamic Programming

Many DPs cannot be solved exactly due to the “curse of dimensionality”. However, many options exist to find “good” solutions:

- Heuristics
- State aggregation
- Basis function approximations

The ideal approach is problem dependent.

Dimension Reduction Methods

Basis function approximations approximate the optimal value function as a weighted sum of basis functions:

For certain types of functions the value function can be expressed exactly → $V(s) = \sum_{k=1}^{\infty} r_k \phi_k(s)$

Realistically a finite set of K functions must be selected → $V(s) \approx \tilde{V}(s) = \sum_{k=1}^{K} r_k \phi_k(s)$

1. Select Basis functions $\phi_1, \dots, \phi_K$, which are function of the original state characteristics (e.g. blood pressure, cholesterol)

2. Compute weights $r_1, \dots, r_K$ so that $\sum_{k=1}^{K} r_k \phi_k(s) \approx V(s)$

Basis Function Examples

Linear regression:

$$v(s) \approx r_1 x_1(s) + \cdots + r_n x_n(s)$$

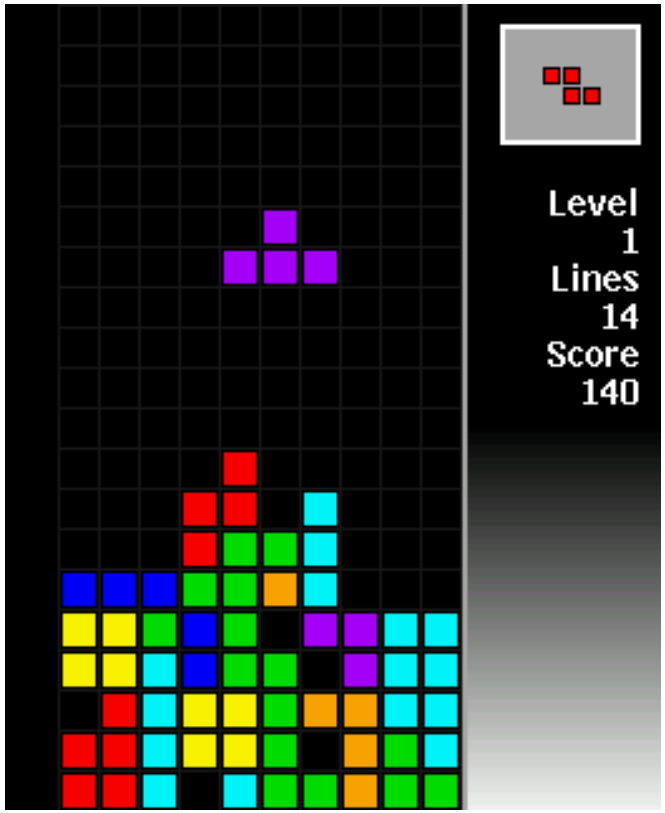
Radial basis functions:

$$v(s) \approx r_1 e^{\beta_1 (x(s)-\mu_1)^2} + r_1 e^{\beta_2 (x(s)-\mu_2)^2} + \cdots + r_n e^{\beta_n (x(s)-\mu_n)^2}$$

Others:

- Polynomials
- Wavelets
- Grid approximations

Example: Tetris



$\phi_1, \ldots, \phi_{10}$: Basis functions mapping the state to the height h_k of each of the ten columns

$\phi_{11}, \ldots, \phi_{19}$: Basis functions mapping the state to the absolute difference between heights of successive columns: $|h_{k+1} - h_k|, k = 1, \ldots, 9$

ϕ_{20}: Basis function maps state to the maximum column height: $\max_k h_k$

ϕ_{21}: Basis function maps state to number of "holes" in the wall

ϕ_{22}: Basis function is equal to one for every state

Approximate Linear Program

The linear programming formulation of an MDP can be used as a way to estimate basis function coefficients.

$$\min \sum_{j \in S} \alpha_j v(j)$$

s.t.

$$v(s) - \lambda \sum_{j \in S} p(j|s,a) v(j) \geq r(s,a), \quad \forall a \in A, s \in S$$

$$v(s) \text{ urs}, \quad \forall s \in S$$

Replacing $v(j)$ with $\tilde{v}(j) = \sum_{k=1}^{K} r_k \phi_k(s)$

$$\min \sum_{j \in S} \alpha_j \sum_{k=1}^{K} r_k \phi_k(j)$$

s.t.

$$\sum_{k=1}^{K} r_k \phi_k(s) - \lambda \sum_{j \in S} p(j|s,a) \sum_{k=1}^{K} r_k \phi_k(s) \geq r(s,a), \quad \forall a \in A, s \in S$$

$$r_k \text{ urs}, \quad \forall k = 1, \ldots, K$$

How good are these approximations?

Example: Tetris

- Using a linear program to set the basis function coefficient resulted in policies with an average of 4274 points per game (human experts average about 3200 points per game)
- Other more advanced methods have lead to improvements averaging 5500 points per game

Where to Learn More

- Schweitzer PJ, Seidmann A. Generalized Polynomial Approximations in Markovian Decision Processes. Journal of Mathematical Analysis and Applications. 1985;110:568–582
- De Farias DP, Van Roy B. The Linear Programming Approach to Approximate Dynamic Programming. Operations Research. 2003;51(6):850–865
- Powell WB. Approximate Dynamic Programming: Solving the Curses of Dimensionality. Wiley; 2007

Interesting healthcare example:

- Patrick, Jonathan, Martin L. Puterman, and Maurice Queyranne. "Dynamic multipriority patient scheduling for a diagnostic resource." *Operations research* 56, no. 6 (2008): 1507-1525.

Model-Free Methods

Two major sources of challenges to solving MDPs are:

1) "curse of dimensionality"
2) "curse of modeling"

"Model-Free" methods are suited to problems of type 2, for which transition probabilities are not known

These methods are known under various names including: ***reinforcement learning***

Model-Free Methods

Monte Carlo sampling is a common approach for estimating the expectation of functions of random variables

Model free approaches use **sample paths** to estimate the value function

These methods are known under various names including: ***reinforcement learning***

Monte-Carlo Sampling

Model free approaches use sample paths to estimate the value function via Monte Carlo sampling

$$E^{\pi}\left[\sum_{t=1}^{N-1} r_t(s_t, \pi(s_t)) + r_N(s_N)\right]$$

$$\approx \frac{1}{K}\sum_{k=1}^{K}\sum_{t=1}^{N-1} r_t\left(s_t^k, \pi(s_t^k)\right) + r_N\left(s_N^k\right)]$$

Where $k = 1, \ldots, K$ are random sample paths from the Markov chain.

Monte Carlo Policy Evaluation

A selected policy π can be evaluated approximately via Monte Carlo sampling

As $\mathrm{K} \to \infty$ $\tilde{v}^{\pi}(s_0) \to v^{\pi}(s_0)$.

In practice the number of samples, N, must be chosen to tradeoff between (a) some desired level of confidence and (b) a computational budget.

Example: Bandit Problem

Consider a game in which your friend holds two coins: 1 coin is fair, the other is biased towards landing heads up.

You know your friend holds two different coins but you don't know the likelihood of each turning up a head.

Each turn you get to select the coin your friend will flip. If you win you get $1 if you lose you lose $1.

Question: how would you play this game?

Application: medical treatment decisions with multiple treatment options and uncertain rewards

Example: multi-armed bandit

The action is which "arm", a, to try at each decision epoch, and the expected reward for this action is $Q_t(a)$.

Since $Q_t(a)$ is not known exactly it must be estimated as:

$$\widetilde{Q_t}(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}$$

Where k_a is the number of times arm a has been pulled.

As $k_a \to \infty \ \widetilde{Q_t}(a) \to Q_t(a),$ thus sampling each arm an infinite number of times will identify the optimal action

$$a^* = argmax_{a \in A}\{Q_t(a)\}.$$

Example: multi-armed bandit

Policies obtained from learning attempt to converge to a near optimal policy quickly

The simplest learning-based policy is the greedy policy:

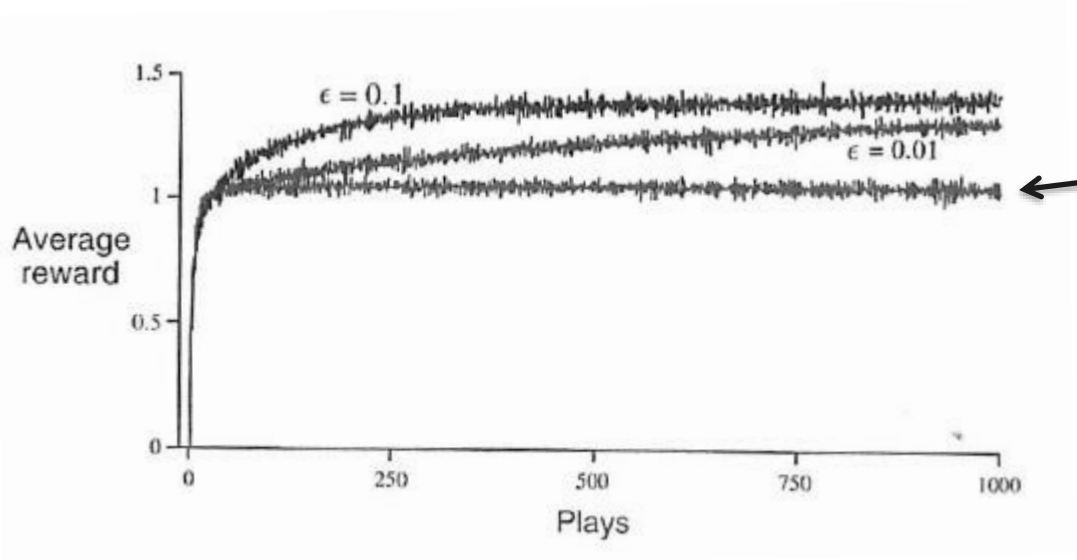
$$\tilde{a} = argmax\{\widetilde{Q_t}(a)\}$$

Alternatively the $\epsilon - greedy$ method explores the action set by randomly selecting actions with probability ϵ

As $\mathrm{k_a} \to \infty$ $Q_t(a) \to Q_t^*(a)$ and the optimal action is selected with probability greater than $1 - \epsilon$.

Example: multi-armed bandit

The following results are averages across 2000 randomly generated 10-armed bandit problems for 3 different greedy methods.



Taken from “Reinforcement Learning: An Introduction”, By Sutton and Barto, MIT Press

Monte Carlo Policy Iteration

For more complex problems with multiple system states the following algorithm can be used

Algorithm (MC Policy Iteration):

1. For all s initialize $\pi(\mathrm{s})$ and $\mathrm{Q}(\mathrm{s}, \pi(s))$. Choose a suitably large N.

2. ***Policy Evaluation***:

   Randomly select a starting pair, $(\mathrm{s}, \pi(s))$, and generate a sample path of length N

   For all $(\mathrm{s}, \pi(s))$ in the sample path compute: $\tilde{Q}^{\pi}(s, \pi(s)) = \sum_{t=n_s}^{N-1} \lambda^t r_t(s_t, \pi(s_t)) + \lambda^N r_N(s_N)$, where n_s is the index for the first instance state s is encountered.

3. ***Policy Improvement***:

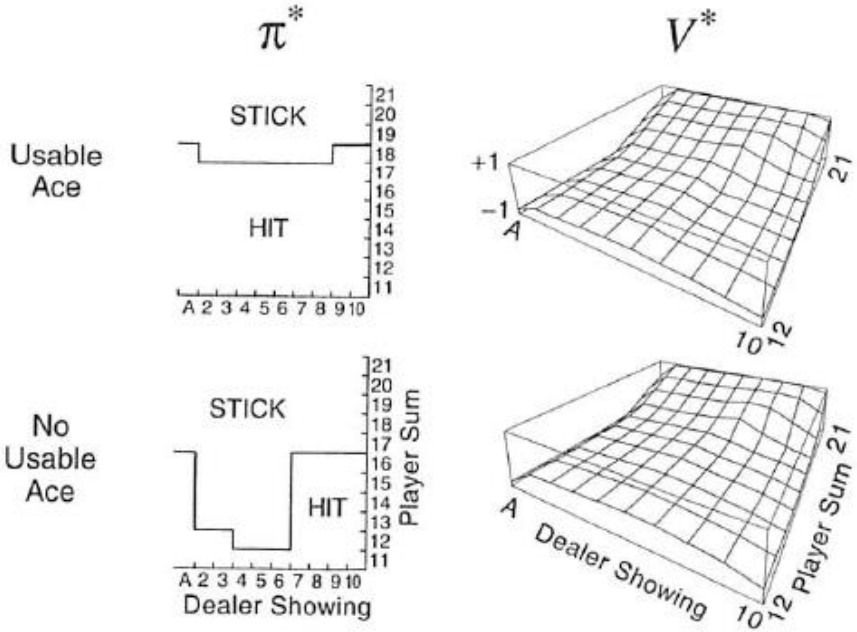
   For all s: $\pi(\mathrm{s}) \in \mathrm{argmax}_{\mathrm{a} \in \mathrm{A}}\{\mathrm{Q}(\mathrm{s}, \mathrm{a})\}$

   Return to Step 2;

Example: Blackjack

Black jack is a sequential card game with a "player" and a "dealer". The player receives two cards face up; the dealer one card face down and one face up. The player must decide one at a time whether to take a card (hit) or not (stick). The goal is to get as close to 21 without going over.

Results from Q-learning:



Taken from "Reinforcement Learning: An Introduction", By Sutton and Barto, MIT Press

Incremental updating

The Robins-Monro Algorithm is a recency-weighted updating approach to estimate the expectation of random variables incrementally.

Algorithm (Robins-Monro):

1. Let $\mathrm{X_i}$ denote the i^{th} sample of random variable X, and let $\mathrm{Y^m}$ denote the estimate of $\mathrm{E[X]}$ in the $\mathrm{m^{th}}$ iteration; Set $\mathrm{Y^0} = 0$.

2. Update Y as follows: $\mathrm{Y^m} = (1 - \mu_m)Y^{m-1} + \mu_m X_m$

3. If $|\mathrm{Y^m} - \mathrm{Y^{m-1}}| < \epsilon$ then stop; Otherwise $\mathrm{m} = \mathrm{m} + 1$ and return to Step 2

The parameter $\mu_m \in [0,1]$ is the **step size** and it defines the relative weight of the current estimate and most recent sample.

Q-Learning

Q-Learning uses the Robins-Monro algorithm to estimate the **Q-value** of a given state action pair (s, a), defined as:

$$Q^{\pi}(s,a) = r(s,a) + \sum_{j=1}^{N} p(j|i,a)\, v^{\pi}(j)$$

$Q(s,a)$ is estimated by sampling as follows:

$$Q_m(s,a) = (1-\mu_m)Q_{m-1}(s,a) + \mu_m(r(s,a) + \lambda \max_{a' \in A} Q_{m-1}(s,a'))$$

Q-Learning

Algorithm (Q-Learning):

1. Set $\mathrm{Q}(\mathrm{s},\mathrm{a}) = 0$ for all $\mathrm{s} \in S$ and $\mathrm{a} \in A$. Set $\mathrm{m} = 0$ and max iterations K.

2. Let the current state be i. With probability ϵ select action, a, randomly; with probability $1 - \epsilon$ choose $a = \mathrm{argmax}\{\mathrm{Q}_m(\mathrm{i},\mathrm{a})\}$

3. Sample a new state, j, given action a, and update $\mathrm{Q_m}(\mathrm{i},\mathrm{a})$ as

$$Q_m(i,a) = (1-\mu_m)Q_{m-1}(i,a) + \mu_m\left(r(j,a) + \lambda \max_{a' \in A} Q_{m-1}(j,a')\right)$$

4. If $\mathrm{m} < \mathrm{K}$ set $\mathrm{i} = j$ and return to Step 2. Otherwise go to Step 5.

5. For each state, s, choose the action, a, that maximizes $\mathrm{Q^K}(\mathrm{s},\mathrm{a})$

Where to Learn More

Abhijit Gosavi, 2009, Reinforcement Learning: A Tutorial Survey and Recent Advances, *INFORMS Journal on Computing*, 212, 178-192.

“Reinforcement Learning: An Introduction”, By Sutton and Barto, MIT Press

Sections

- Finite Horizon MDPs 
- Infinite Horizon MDPs 
- Special Topics 
  - Partially observable MDPs
  - Robust MDPs
  - Approximate DP
  - Reinforcement Learning

Acknowledgements

Oguz Alagoz (University of Wisconsin)
Murat Kurt (Merck)
Jennifer Mason (University of Virginia)
Nilay Shah (Mayo Clinic)
Lauren Steimle (University of Michigan)
Yuanhui Zhang (Research Triangle Institute)

This work was funded in part by grants CMMI-1536444 and CMMI-1462060 from the Operations Engineering program at the *National Science Foundation.*

COLLEGE OF ENGINEERING
INDUSTRIAL & OPERATIONS ENGINEERING
UNIVERSITY OF MICHIGAN

Brian Denton
Industrial and Operations Engineering
University of Michigan

btdenton@umich.edu

The slides (and pictures!) are on my website:

http://umich.edu/~btdenton