Non-Deterministic (Probabilistic) Planning

Blai Bonet

Universidad Simón Bolívar

Summer School – ICAPS 2011 – Freiburg, Germany

Objectives

Main goal is to learn about:

- underlying mathematical models
- standard algorithms such as value and policy iteration
- search-based algorithms such as LAO* and RTDP
- representation languages and heuristics

Objectives

Main goal is to learn about:

- underlying mathematical models
- standard algorithms such as value and policy iteration
- search-based algorithms such as LAO* and RTDP
- representation languages and heuristics

We will also learn about:

- mathematical notation and techniques
- open problems and research opportunities
- pointers and references to related work (at the end)

Outline of the Lecture

Part I: Background

Part II: Models

Part III: Algorithms

Part IV: Languages

Part V: Heuristics

Part VI: Variants and Extensions

Part I

Background

Probability: Expectation

Let X be a random variable, $A \subseteq \mathbb{R}$ and $f(\cdot)$ a function

- $E[f(X); A] = \sum_{x \in A} f(x) \cdot P(X = x)$
- $E[f(X)] = E[f(X); A] + E[f(X); A^c]$
- $P(X \in A) = E[1_A] = E[1; A]$
- $P(X \in A) = 0 \implies E[f(X)] = E[f(X); A]$

Probability: Conditional Expectation

Let Z and X be random variables, and $f(\cdot)$ a function

Conditional expectation of $f(Z)$ given X is $E[f(Z)|X]$

Probability: Conditional Expectation

Let Z and X be random variables, and $f(\cdot)$ a function

Conditional expectation of $f(Z)$ given X is $E[f(Z)|X]$

It is a **function** $F(\cdot)$ that depends on X; i.e.

- $F(x) = E[f(Z)|X = x]$

Probability: Conditional Expectation

Let Z and X be random variables, and $f(\cdot)$ a function

Conditional expectation of $f(Z)$ given X is $E[f(Z)|X]$

It is a **function** $F(\cdot)$ that depends on X; i.e.

- $F(x) = E[f(Z)|X = x]$

Thm (Iterated Exp.): let Y be a random variable. Then,

$$E[f(Z)|X] = E[\underbrace{E[f(Z)|X,Y]}_{\text{func. of } X,Y}|X]$$

Probability: Conditional Expectation

Let Z and X be random variables, and $f(\cdot)$ a function

Conditional expectation of $f(Z)$ given X is $E[f(Z)|X]$

It is a **function** $F(\cdot)$ that depends on X; i.e.

- $F(x) = E[f(Z)|X = x]$

Thm (Iterated Exp.): let Y be a random variable. Then,

$$E[f(Z)|X] = E[\underbrace{E[f(Z)|X, Y]}_{\text{func. of } X, Y}|X]$$

Cor: $E[f(Z)] = E[E[f(Z)|X]]$

Probability: Markov Chains

Elements:

- finite state space S
- **timed-indexed transition probabilities** $p_i(s|s')$

Unique probabilities $\{P_s\}_s$ over trajectories $\Omega = S^\infty$ such that

$$P_s(\langle s_0, s_1, s_2, \ldots, s_{n+1}\rangle) = \\ [\![s = s_0]\!] \cdot p_0(s_1|s_0) \cdot p_1(s_2|s_1) \cdots p_n(s_{n+1}|s_n)$$

Probability: Markov Chains

Elements:

- finite state space S
- **timed-indexed transition probabilities** $p_i(s|s')$

Unique probabilities $\{P_s\}_s$ over trajectories $\Omega = S^\infty$ such that

$$P_s(\langle s_0, s_1, s_2, \ldots, s_{n+1}\rangle) = \\ [\![s = s_0]\!] \cdot p_0(s_1|s_0) \cdot p_1(s_2|s_1) \cdots p_n(s_{n+1}|s_n)$$

Expectation wrt P_s is denoted as E_s

Probability: Markov Property

If $p_i = p_j$ for all i, j, the chain is **time homogeneous** (TH)

A time-homogenous chain satisfies the **Markov property**:

for any function f over trajectories:

$$\underbrace{E_s[f(X_k, X_{k+1}, \ldots)|X_1, \ldots, X_k]}_{\text{func. of } X_1, \ldots, X_k} = \underbrace{E_{X_k}[f(X)]}_{\text{func. of } X_k}$$

Probability: Markov Property

If $p_i = p_j$ for all i, j, the chain is **time homogeneous** (TH)

A time-homogenous chain satisfies the **Markov property**:

for any function f over trajectories:

$$\underbrace{E_s[f(X_k, X_{k+1}, \ldots)|X_1, \ldots, X_k]}_{\text{func. of } X_1, \ldots, X_k} = \underbrace{E_{X_k}[f(X)]}_{\text{func. of } X_k}$$

For $k = 1$,

$$E_s[f(X_1, X_2, \ldots)|X_1] = E_{X_1}[f(X)]$$

Operators

An operator maps functions to functions. We will use simple operators that map vectors in $\mathbb{R}_+^N$ to vectors in $\mathbb{R}_+^N$

Extended non-negative reals $\overline{\mathbb{R}}_+ = [0, \infty) \cup \{\infty\}$

Let $T : \overline{\mathbb{R}}_+^N \to \overline{\mathbb{R}}_+^N$ be an operator:

- **monotone**: $J \leq J'$ implies $TJ \leq TJ'$
- **continuous**: J_k is a monotone sequence that converges to J implies $TJ_k \to TJ$
- **fixed points**: J is a fixed point of T iff $TJ = J$

Fixed Points (a la Tarski)

Thm: Let T be a **monotone** operator and $\mathcal{FP}$ the set of fixed points of T. Then

- $\mathcal{FP}$ is non-empty
- $\mathcal{FP}$ has minimum (LFP) and maximum (GFP) (wrt pw $\leq$)
- if T is continuous, the LFP is $\lim_{k\to\infty} T^k 0$

Fixed Points (a la Tarski)

Thm: Let T be a **monotone** operator and $\mathcal{FP}$ the set of fixed points of T. Then

- $\mathcal{FP}$ is non-empty
- $\mathcal{FP}$ has minimum (LFP) and maximum (GFP) (wrt pw $\leq$)
- if T is continuous, the LFP is $\lim_{k\to\infty} T^k 0$

Remark: Thm is more general. It holds for any **monotone** function on a **complete** lattice. It says that $\mathcal{FP}$ is a complete lattice as well

Part II

Models

Goals of Part II

- Understand the underlying model for probabilistic planning
- Understand the solutions for these models
- Familiarity with notation and formal methods
- Knowledge about basic facts

Models for Classical Planning

Characterized by:

- A finite **state space** S
- a finite set of **actions** A with subsets $A(s) \subseteq A$ of actions applicable at each state $s \in S$
- a **deterministic transition function** $f : S \times A \to S$ such that $f(s, a)$ is the state that results of applying action $a \in A(s)$ in state $s \in S$
- **initial state** $s_{init} \in S$
- a subset $G \subseteq S$ of **goal states**
- **positive costs** $c : S \times A \to \mathbb{R}^+$ where $c(s, a)$ is the cost of applying action $a \in A(s)$ in state $s \in S$ (often, $c(s, a)$ only depends on s; i.e. costs are given as $c : A \to \mathbb{R}^+$)

Solutions for Classical Planning

A plan is a sequence $\pi = \langle a_0, a_1, \ldots, a_n \rangle$ of actions that defines a **trajectory** $s_0, s_1, \ldots, s_{n+1}$ where

- $s_0 = s_{init}$ is the initial state
- $a_i \in A(s_i)$ is an applicable action at state s_i, $i = 0, \ldots, n$
- $s_{i+1} = f(s_i, a_i)$ is the result of applying action a_i at state s_i

The plan π is a **solution** iff s_{n+1} is a goal state

The **cost** of π is $c(s_0, a_0) + c(s_1, a_1) + \cdots + c(s_n, a_n)$

A solution is **optimal** if it is of minimum cost

Actions with Uncertain Effects

Certain problems contains actions whose intrinsic behaviour is **non-deterministic**. For example, tossing a coin or rolling a dice are actions whose outcomes cannot be predicted with certainty

In other cases, uncertainty is the result of a **coarse model** that does not include all the information required to predict the outcomes of actions

In both cases, it is convenient to consider problems with **non-deterministic actions**

Extending the Classical Model into the Non-Deterministic Model

- A finite **state space** S
- a finite set of **actions** A with subsets $A(s) \subseteq A$ of actions applicable at each state $s \in S$
- a **non-deterministic transition function** $F : S \times A \to 2^S$ such that $F(s,a)$ is the set of states that **may** result of applying action $a \in A(s)$ in state $s \in S$ (with $F(s,a) \neq \emptyset$)
- **initial state** $s_{init} \in S$
- a subset $G \subseteq S$ of **goal states**
- **positive costs** $c : S \times A \to \mathbb{R}^+$ where $c(s,a)$ is the cost of applying action $a \in A(s)$ in state $s \in S$

States are assummed to be fully observable

Probabilistic Planning Problems

- A finite **state space** S
- a finite set of **actions** A with subsets $A(s) \subseteq A$ of actions applicable at each state $s \in S$
- **probabilities** $p(s'|s,a)$ that express the probability of reaching state s' when action $a \in A(s)$ is applied at $s \in S$
- **initial state** $s_{init} \in S$
- a subset $G \subseteq S$ of **goal states**
- **positive costs** $c : S \times A \rightarrow \mathbb{R}^+$ where $c(s,a)$ is the cost of applying action $a \in A(s)$ in state $s \in S$

States are assummed to be fully observable

Solutions

A solution cannot be a sequence of actions because the outcomes of the actions **cannot be predicted**

However, since states are fully observable, we can think on **contingent plans**

Contingent Plans

Many ways to formalize contingent plans. The most general corresponds to **functions** that map states into actions

A contingent plan is a sequence $\pi = \langle \mu_0, \mu_1, \ldots \rangle$ of **decision functions** $\mu_i : S \rightarrow A$ such that if at time i the **current state** is $s \in S$, then the action to execute is $\mu_i(s)$

Contingent Plans

Many ways to formalize contingent plans. The most general corresponds to **functions** that map states into actions

A contingent plan is a sequence $\pi = \langle \mu_0, \mu_1, \ldots \rangle$ of **decision functions** $\mu_i : S \rightarrow A$ such that if at time i the **current state** is $s \in S$, then the action to execute is $\mu_i(s)$

A contingent plan generates a **set of trajectories** $Traj_\pi(s)$ from an initial state s: $\langle s_0, s_1, s_2, \ldots \rangle \in Traj_\pi(s)$ iff

- $s_0 = s$
- $s_{i+1} \in F(s_i, \mu_i(s_i))$ and $\mu_i(s_i) \in A(s_i)$
- if $s_i \in G$, then $s_{i+1} = s_i$ (convenient)

Cost of Plans (Intuition)

Each trajectory $\tau = \langle s_0, s_1, \ldots \rangle$ has **probability**

$$P(\tau) = p(s_1|s_0, \mu_0(s_0)) \cdot p(s_2|s_1, \mu_1(s_1)) \cdots$$

where $p(s|s,a) = 1$ for all $a \in A$ when $s \in G$ (convenient)

Cost of Plans (Intuition)

Each trajectory $\tau = \langle s_0, s_1, \ldots \rangle$ has **probability**

$$P(\tau) = p(s_1|s_0, \mu_0(s_0)) \cdot p(s_2|s_1, \mu_1(s_1)) \cdots$$

where $p(s|s,a) = 1$ for all $a \in A$ when $s \in G$ (convenient)

Each trajectory has **cost**

$$c(\tau) = c(s_0, \mu_0(s_0)) + c(s_1, \mu_1(s_1)) + \cdots$$

where $c(s,a) = 0$ for all $a \in A$ and $s \in G$ (convenient)

Cost of Plans (Intuition)

Each trajectory $\tau = \langle s_0, s_1, \ldots \rangle$ has **probability**

$$P(\tau) = p(s_1|s_0, \mu_0(s_0)) \cdot p(s_2|s_1, \mu_1(s_1)) \cdots$$

where $p(s|s,a) = 1$ for all $a \in A$ when $s \in G$ (convenient)

Each trajectory has **cost**

$$c(\tau) = c(s_0, \mu_0(s_0)) + c(s_1, \mu_1(s_1)) + \cdots$$

where $c(s,a) = 0$ for all $a \in A$ and $s \in G$ (convenient)

The **cost of policy** π at state s is

$$J_\pi(s) = \sum_\tau c(\tau) \cdot P(\tau)$$

Cost of Plans (Formal)

Under **fixed** π, the system becomes a **Markov chain**

with transition probabilities $p_i(s'|s) = p(s'|s, \mu_i(s))$

The transitions induce **probabilities** $\{P^\pi_s\}_s$ over trajectories $S^\infty = \{\langle s_0, s_1, \ldots \rangle | s_i \in S\}$ and **expectations** $\{E^\pi_s\}_s$

Cost of Plans (Formal)

Under **fixed** π, the system becomes a **Markov chain**

with transition probabilities $p_i(s'|s) = p(s'|s, \mu_i(s))$

The transitions induce **probabilities** $\{P_s^\pi\}_s$ over trajectories $S^\infty = \{\langle s_0, s_1, \ldots\rangle | s_i \in S\}$ and **expectations** $\{E_s^\pi\}_s$

Let X_i be the r.v. that denotes the state of the chain at time i
The **cost of policy** π at state s is defined as

$$J_\pi(s) = E_s^\pi\left[\sum_{i=0}^{\infty} c(X_i, \mu_i(X_i))\right]$$

Recursion I

Policy π is **stationary** (i.e. decisions do not depend on time) if $\mu = \mu_i$ for all $i \geq 0$. Stationary policy denoted as μ

Under μ, the chain is TH and satisfies the Markov property

Recursion I

Policy π is **stationary** (i.e. decisions do not depend on time) if $\mu = \mu_i$ for all $i \geq 0$. Stationary policy denoted as μ

Under μ, the chain is TH and satisfies the Markov property

If μ is stationary then

$$J_\mu(s)$$

Recursion I

Policy π is **stationary** (i.e. decisions do not depend on time) if $\mu = \mu_i$ for all $i \geq 0$. Stationary policy denoted as μ

Under μ, the chain is TH and satisfies the Markov property

If μ is stationary then

$$J_\mu(s) = E_s^\mu\left[c(X_0, \mu(X_0)) + \textstyle\sum_{i=1}^\infty c(X_i, \mu(X_i))\right]$$

Recursion I

Policy π is **stationary** (i.e. decisions do not depend on time) if $\mu = \mu_i$ for all $i \geq 0$. Stationary policy denoted as μ

Under μ, the chain is TH and satisfies the Markov property

If μ is stationary then

$$J_\mu(s) = E_s^\mu\left[c(X_0, \mu(X_0)) + \textstyle\sum_{i=1}^\infty c(X_i, \mu(X_i))\right]$$

$$= c(s, \mu(s)) + E_s^\mu\left[\textstyle\sum_{i=1}^\infty c(X_i, \mu(X_i))\right]$$

Recursion I

Policy π is **stationary** (i.e. decisions do not depend on time) if $\mu = \mu_i$ for all $i \geq 0$. Stationary policy denoted as μ

Under μ, the chain is TH and satisfies the Markov property

If μ is stationary then

$$
\begin{aligned}
J_\mu(s) &= E^\mu_s\left[c(X_0, \mu(X_0)) + \textstyle\sum_{i=1}^\infty c(X_i, \mu(X_i))\right] \\
&= c(s, \mu(s)) + E^\mu_s\left[\textstyle\sum_{i=1}^\infty c(X_i, \mu(X_i))\right] \\
&= c(s, \mu(s)) + E^\mu_s\left[E^\mu_s\left[\textstyle\sum_{i=1}^\infty c(X_i, \mu(X_i)) \middle| X_1\right]\right]
\end{aligned}
$$

Recursion I

Policy π is **stationary** (i.e. decisions do not depend on time) if $\mu = \mu_i$ for all $i \geq 0$. Stationary policy denoted as μ

Under μ, the chain is TH and satisfies the Markov property

If μ is stationary then

$$
\begin{aligned}
J_\mu(s) &= E_s^\mu\left[c(X_0, \mu(X_0)) + \textstyle\sum_{i=1}^\infty c(X_i, \mu(X_i))\right] \\
&= c(s, \mu(s)) + E_s^\mu\left[\textstyle\sum_{i=1}^\infty c(X_i, \mu(X_i))\right] \\
&= c(s, \mu(s)) + E_s^\mu\left[E_s^\mu\left[\textstyle\sum_{i=1}^\infty c(X_i, \mu(X_i)) \middle| X_1\right]\right] \\
&= c(s, \mu(s)) + E_s^\mu\left[E_{X_1}^\mu\left[\textstyle\sum_{i=0}^\infty c(X_i, \mu(X_i))\right]\right]
\end{aligned}
$$

Recursion I

Policy π is **stationary** (i.e. decisions do not depend on time) if $\mu = \mu_i$ for all $i \geq 0$. Stationary policy denoted as μ

Under μ, the chain is TH and satisfies the Markov property

If μ is stationary then

$$
\begin{aligned}
J_\mu(s) &= E_s^\mu \left[c(X_0, \mu(X_0)) + \textstyle\sum_{i=1}^\infty c(X_i, \mu(X_i)) \right] \\
&= c(s, \mu(s)) + E_s^\mu \left[\textstyle\sum_{i=1}^\infty c(X_i, \mu(X_i)) \right] \\
&= c(s, \mu(s)) + E_s^\mu \left[E_s^\mu \left[\textstyle\sum_{i=1}^\infty c(X_i, \mu(X_i)) \middle| X_1 \right] \right] \\
&= c(s, \mu(s)) + E_s^\mu \left[E_{X_1}^\mu \left[\textstyle\sum_{i=0}^\infty c(X_i, \mu(X_i)) \right] \right] \\
&= c(s, \mu(s)) + \textstyle\sum_{s'} p(s'|s, \mu(s)) E_{s'}^\mu \left[\textstyle\sum_{i=0}^\infty c(X_i, \mu(X_i)) \right]
\end{aligned}
$$

Recursion I

Policy π is **stationary** (i.e. decisions do not depend on time) if $\mu = \mu_i$ for all $i \geq 0$. Stationary policy denoted as μ

Under μ, the chain is TH and satisfies the Markov property

If μ is stationary then

$$\begin{aligned} J_\mu(s) &= E^\mu_s\left[c(X_0,\mu(X_0)) + \textstyle\sum_{i=1}^\infty c(X_i,\mu(X_i))\right] \\ &= c(s,\mu(s)) + E^\mu_s\left[\textstyle\sum_{i=1}^\infty c(X_i,\mu(X_i))\right] \\ &= c(s,\mu(s)) + E^\mu_s\left[E^\mu_s\left[\textstyle\sum_{i=1}^\infty c(X_i,\mu(X_i))\middle|X_1\right]\right] \\ &= c(s,\mu(s)) + E^\mu_s\left[E^\mu_{X_1}\left[\textstyle\sum_{i=0}^\infty c(X_i,\mu(X_i))\right]\right] \\ &= c(s,\mu(s)) + \textstyle\sum_{s'} p(s'|s,\mu(s)) E^\mu_{s'}\left[\textstyle\sum_{i=0}^\infty c(X_i,\mu(X_i))\right] \end{aligned}$$

I.e., $J_\mu(s) = c(s,\mu(s)) + \sum_{s'} p(s'|s,\mu(s)) J_\mu(s')$

Solutions

Policy π is **solution for state** s if π reaches a goal **with probability 1** from state s

A policy π is **solution** if it is solution for each s

Solutions

Policy π is **solution for state** s if π reaches a goal **with probability 1** from state s

A policy π is **solution** if it is solution for each s

In prob. planning, we are interested in solutions for s_{init}

Proper Policies

Let μ be a stationary policy. We say that μ is **proper** if

$$\max_s P_s^\mu(X_N \notin G) = \rho_\mu < 1$$

where $N = |S|$ is the number of states

Properties of Stationary Solutions 1/4

Let $T = \min\{i : X_i \in G\}$ be the **time to arrive to goal**

- μ is a solution for s iff $P_s^\mu(T = \infty) = 0$
- μ is proper iff $P_s^\mu(T > N) < 1$ for all s

Properties of Stationary Solutions 1/4

Let $T = \min\{i : X_i \in G\}$ be the **time to arrive to goal**

- μ is a solution for s iff $P_s^\mu(T = \infty) = 0$
- μ is proper iff $P_s^\mu(T > N) < 1$ for all s

Let $R^\mu(s)$ be the set of **reachable states from** s **using** μ

Lemma: if $P_{s'}^\mu(T > N) < 1$ for all $s' \in R^\mu(s)$, $E_s^\mu T < \infty$

Lemma: if $P_s^\mu(T = \infty) = 0$, $P_{s'}^\mu(T > N) < 1$ for $s' \in R^\mu(s)$

Lemma: $P_s^\mu(T > m) \leq P_s^\mu(T > n)$ for all $n < m$

Properties of Stationary Solutions 2/4

Thm: μ is a solution for s iff $E^{\mu}_{s}T < \infty$

Skip Proof

Properties of Stationary Solutions 2/4

Thm: μ is a solution for s iff $E_s^\mu T < \infty$

Proof:
$(\Leftarrow)$

$$E_s^\mu T < \infty \implies P_s^\mu(T = \infty) = 0$$

Properties of Stationary Solutions 2/4

Thm: μ is a solution for s iff $E_s^\mu T < \infty$

Proof:
($\Rightarrow$) Note: μ sol. for $s \implies P_s^\mu(T = \infty) = 0 \implies$ Lemma 2

$$P_s^\mu(T > N(k+1)) = E_s^\mu[P_s^\mu(T > N(k+1)|X_0, \ldots, X_{Nk})]$$

Properties of Stationary Solutions 2/4

Thm: μ is a solution for s iff $E_s^\mu T < \infty$

Proof:

($\Rightarrow$) Note: μ sol. for $s \implies P_s^\mu(T = \infty) = 0 \implies$ Lemma 2

$$P_s^\mu(T > N(k+1)) = E_s^\mu[P_s^\mu(T > N(k+1) | X_0, \ldots, X_{Nk})]$$

$$= E_s^\mu[P_{X_{Nk}}^\mu(T > N)]$$

Properties of Stationary Solutions 2/4

Thm: μ is a solution for s iff $E_s^\mu T < \infty$

Proof:

($\Rightarrow$) Note: μ sol. for $s \implies P_s^\mu(T = \infty) = 0 \implies$ Lemma 2

$$
\begin{aligned}
P_s^\mu(T > N(k+1)) &= E_s^\mu[P_s^\mu(T > N(k+1) | X_0, \ldots, X_{Nk})] \\
&= E_s^\mu[P_{X_{Nk}}^\mu(T > N)] \\
&= E_s^\mu[P_{X_{Nk}}^\mu(T > N)\,;\, X_{Nk} \notin G]
\end{aligned}
$$

Properties of Stationary Solutions 2/4

Thm: μ is a solution for s iff $E^\mu_s T < \infty$

Proof:

($\Rightarrow$) Note: μ sol. for $s \implies P^\mu_s(T = \infty) = 0 \implies$ Lemma 2

$$\begin{aligned} P^\mu_s(T > N(k+1)) &= E^\mu_s[P^\mu_s(T > N(k+1) | X_0, \ldots, X_{Nk})] \\ &= E^\mu_s[P^\mu_{X_{Nk}}(T > N)] \\ &= E^\mu_s[P^\mu_{X_{Nk}}(T > N)\,;\, X_{Nk} \notin G] \\ &\leq P^\mu_s(T > Nk) \cdot \max_{s' \in R^\mu(s)} P^\mu_{s'}(T > N) \end{aligned}$$

Properties of Stationary Solutions 2/4

Thm: μ is a solution for s iff $E_s^\mu T < \infty$

Proof:

($\Rightarrow$) Note: μ sol. for $s \implies P_s^\mu(T = \infty) = 0 \implies$ Lemma 2

$$
\begin{aligned}
P_s^\mu(T > N(k+1)) &= E_s^\mu[P_s^\mu(T > N(k+1)|X_0, \ldots, X_{Nk})] \\
&= E_s^\mu[P_{X_{Nk}}^\mu(T > N)] \\
&= E_s^\mu[P_{X_{Nk}}^\mu(T > N)\,;\, X_{Nk} \notin G] \\
&\leq P_s^\mu(T > Nk) \cdot \max_{s' \in R^\mu(s)} P_{s'}^\mu(T > N) \\
&\leq \rho^k \cdot \rho = \rho^{k+1} \qquad \text{(some } \rho < 1 \text{ by Lemma 2)}
\end{aligned}
$$

Properties of Stationary Solutions 2/4

Thm: μ is a solution for s iff $E_s^\mu T < \infty$

Proof:
($\Rightarrow$) Note: μ sol. for $s \implies P_s^\mu(T = \infty) = 0 \implies$ Lemma 2

$$\begin{aligned}
P_s^\mu(T > N(k+1)) &= E_s^\mu[P_s^\mu(T > N(k+1) | X_0, \ldots, X_{Nk})] \\
&= E_s^\mu[P_{X_{Nk}}^\mu(T > N)] \\
&= E_s^\mu[P_{X_{Nk}}^\mu(T > N)\,;\, X_{Nk} \notin G] \\
&\leq P_s^\mu(T > Nk) \cdot \max_{s' \in R^\mu(s)} P_{s'}^\mu(T > N) \\
&\leq \rho^k \cdot \rho = \rho^{k+1} \qquad \text{(some } \rho < 1 \text{ by Lemma 2)}
\end{aligned}$$

$$E_s^\mu T = \sum_{k=0}^{\infty} P_s^\mu(T > k) \overset{\text{Lemma 3}}{\leq} \sum_{k=0}^{\infty} N P_s^\mu(T > Nk) \leq \sum_{k=0}^{\infty} N \rho^k < \infty$$

Properties of Stationary Solutions 3/4

Thm: μ is a solution for s iff $J_\mu(s) < \infty$

Skip Proof

Properties of Stationary Solutions 3/4

Thm: μ is a solution for s iff $J_\mu(s) < \infty$

Proof:

$$E^\mu_s T = \sum_{k=0}^{\infty} k P^\mu_s(T = k) + \infty \cdot P^\mu_s(T = \infty)$$

Properties of Stationary Solutions 3/4

Thm: μ is a solution for s iff $J_\mu(s) < \infty$

Proof:

$$E_s^\mu T = \sum_{k=0}^{\infty} k P_s^\mu(T = k) + \infty \cdot P_s^\mu(T = \infty)$$

$$J_\mu(s) = E_s^\mu \left[\sum_{i=0}^{\infty} c(X_i, \mu(X_i)) \right]$$

Properties of Stationary Solutions 3/4

Thm: μ is a solution for s iff $J_\mu(s) < \infty$

Proof:

$$E_s^\mu T = \sum_{k=0}^{\infty} k P_s^\mu(T = k) + \infty \cdot P_s^\mu(T = \infty)$$

$$\begin{aligned} J_\mu(s) &= E_s^\mu \left[\sum_{i=0}^{\infty} c(X_i, \mu(X_i)) \right] \\ &= \sum_{k=0}^{\infty} E_s^\mu \left[\sum_{i=0}^{\infty} c(X_i, \mu(X_i))\,;\, T = k \right] + \infty \cdot P_s^\mu(T = \infty) \end{aligned}$$

Properties of Stationary Solutions 3/4

Thm: μ is a solution for s iff $J_\mu(s) < \infty$

Proof:

$$E_s^\mu T = \sum_{k=0}^{\infty} k P_s^\mu(T = k) + \infty \cdot P_s^\mu(T = \infty)$$

$$\begin{aligned}
J_\mu(s) &= E_s^\mu \left[\sum_{i=0}^{\infty} c(X_i, \mu(X_i)) \right] \\
&= \sum_{k=0}^{\infty} E_s^\mu \left[\sum_{i=0}^{\infty} c(X_i, \mu(X_i))\,;\, T = k \right] + \infty \cdot P_s^\mu(T = \infty) \\
&= \sum_{k=0}^{\infty} E_s^\mu \left[\sum_{i=0}^{k-1} c(X_i, \mu(X_i))\,;\, T = k \right] + \infty \cdot P_s^\mu(T = \infty)
\end{aligned}$$

Properties of Stationary Solutions 3/4

Thm: μ is a solution for s iff $J_\mu(s) < \infty$

Proof:

$$E_s^\mu T = \sum_{k=0}^{\infty} k P_s^\mu(T = k) + \infty \cdot P_s^\mu(T = \infty)$$

$$\begin{aligned}
J_\mu(s) &= E_s^\mu \left[\sum_{i=0}^{\infty} c(X_i, \mu(X_i)) \right] \\
&= \sum_{k=0}^{\infty} E_s^\mu \left[\sum_{i=0}^{\infty} c(X_i, \mu(X_i))\,;\, T = k \right] + \infty \cdot P_s^\mu(T = \infty) \\
&= \sum_{k=0}^{\infty} E_s^\mu \left[\sum_{i=0}^{k-1} c(X_i, \mu(X_i))\,;\, T = k \right] + \infty \cdot P_s^\mu(T = \infty) \\
&\leq \sum_{k=0}^{\infty} \overline{c} \cdot k \cdot P_s^\mu(T = k) + \infty \cdot P_s^\mu(T = \infty) = \overline{c} \cdot E_s^\mu T
\end{aligned}$$

Properties of Stationary Solutions 3/4

Thm: μ is a solution for s iff $J_\mu(s) < \infty$

Proof:

$$E_s^\mu T = \sum_{k=0}^{\infty} k P_s^\mu(T = k) + \infty \cdot P_s^\mu(T = \infty)$$

$$\begin{aligned}
J_\mu(s) &= E_s^\mu \left[\sum_{i=0}^{\infty} c(X_i, \mu(X_i)) \right] \\
&= \sum_{k=0}^{\infty} E_s^\mu \left[\sum_{i=0}^{\infty} c(X_i, \mu(X_i))\, ;\, T = k \right] + \infty \cdot P_s^\mu(T = \infty) \\
&= \sum_{k=0}^{\infty} E_s^\mu \left[\sum_{i=0}^{k-1} c(X_i, \mu(X_i))\, ;\, T = k \right] + \infty \cdot P_s^\mu(T = \infty) \\
&\leq \sum_{k=0}^{\infty} \bar{c} \cdot k \cdot P_s^\mu(T = k) + \infty \cdot P_s^\mu(T = \infty) = \bar{c} \cdot E_s^\mu T
\end{aligned}$$

Hence, μ is solution for $s \implies E_s^\mu T < \infty \implies J_\mu(s) < \infty$

Properties of Stationary Solutions 4/4

Thm: μ is proper iff μ is solution

Skip Proof

Properties of Stationary Solutions 4/4

Thm: μ is proper iff μ is solution

Proof:
$(\Rightarrow)$

$$\mu \text{ is proper} \overset{\text{Lemma 1}}{\Longrightarrow} E^\mu_s T < \infty \implies P^\mu_s(T = \infty) = 0$$

for all s. Hence, μ is solution

Properties of Stationary Solutions 4/4

Thm: μ is proper iff μ is solution

Proof:
($\Rightarrow$)

$$\mu \text{ is proper} \overset{\text{Lemma 1}}{\Longrightarrow} E_s^\mu T < \infty \implies P_s^\mu(T = \infty) = 0$$

for all s. Hence, μ is solution

($\Leftarrow$) for all s:

$$\mu \text{ is solution for } s \overset{\text{Def}}{\Longrightarrow} P_s^\mu(T = \infty) = 0 \overset{\text{Lemma 2}}{\Longrightarrow} P_{s'}^\mu(T > N) < 1$$

for all $s' \in R^\mu(s)$. Hence, μ is proper

Optimal Solutions

Solution π is **optimal for** s if $J_\pi(s) \leq J_{\pi'}(s)$ for all policies π'

Solution π is (globally) **optimal** if it is optimal for all states

Optimal Solutions

Solution π is **optimal for** s if $J_\pi(s) \leq J_{\pi'}(s)$ for all policies π'

Solution π is (globally) **optimal** if it is optimal for all states

In probabilistic planning, we are interested in:

- Solutions for s_{init}
- Optimal solutions for s_{init}

Fundamental Operators

For stationary policy μ, define the operator T_μ as

$$\begin{aligned}(T_\mu J)(s) &= c(s,\mu(s)) + \sum_{s'} p(s'|s,\mu(s))J(s') \\ &= c(s,\mu(s)) + E_s^\mu J(X_1)\end{aligned}$$

Fundamental Operators

For stationary policy μ, define the operator T_μ as

$$(T_\mu J)(s) = c(s, \mu(s)) + \sum_{s'} p(s'|s, \mu(s))J(s')$$
$$= c(s, \mu(s)) + E_s^\mu J(X_1)$$

Also, define the operator T as

$$(TJ)(s) = \min_{a \in A(s)} c(s, a) + \sum_{s'} p(s'|s, a)J(s')$$

Assume all value functions satisfy $J(s) = 0$ for $s \in G$

Fixed Points

Operators T_μ and T are **monotone** and **continuous**

Therefore, both have a **unique least fixed points**

Lemma: for policy $\pi = \langle \mu_0, \mu_1, \ldots \rangle$ and $J_0 \equiv 0$,

$$(T_{\mu_0} T_{\mu_1} \cdots T_{\mu_{k-1}} J_0)(s) = E_s^\pi \left[\sum_{i=0}^{k-1} c(X_i, \mu_i(X_i)) \right]$$

Thm: the LFP of T_μ is J_μ

Skip Proof

Fixed Points

Operators T_μ and T are **monotone** and **continuous**

Therefore, both have a **unique least fixed points**

Lemma: for policy $\pi = \langle \mu_0, \mu_1, \ldots \rangle$ and $J_0 \equiv 0$,

$$(T_{\mu_0} T_{\mu_1} \cdots T_{\mu_{k-1}} J_0)(s) = E_s^\pi \left[\sum_{i=0}^{k-1} c(X_i, \mu_i(X_i)) \right]$$

Thm: the LFP of T_μ is J_μ

Proof: Let $J_0 = 0$ and $\hat{J}$ be the LFP of T_μ.

Fixed Points

Operators T_μ and T are **monotone** and **continuous**

Therefore, both have a **unique least fixed points**

Lemma: for policy $\pi = \langle \mu_0, \mu_1, \ldots \rangle$ and $J_0 \equiv 0$,

$$(T_{\mu_0} T_{\mu_1} \cdots T_{\mu_{k-1}} J_0)(s) = E_s^\pi \left[\sum_{i=0}^{k-1} c(X_i, \mu_i(X_i)) \right]$$

Thm: the LFP of T_μ is J_μ

Proof: Let $J_0 = 0$ and $\hat{J}$ be the LFP of T_μ.

By Lemma, $(T_\mu^k J_0)(s) = E_s^\mu \left[\sum_{i=0}^{k-1} c(X_i, \mu(X_i)) \right]$.

Fixed Points

Operators T_μ and T are **monotone** and **continuous**

Therefore, both have a **unique least fixed points**

Lemma: for policy $\pi = \langle \mu_0, \mu_1, \ldots \rangle$ and $J_0 \equiv 0$,

$$(T_{\mu_0} T_{\mu_1} \cdots T_{\mu_{k-1}} J_0)(s) = E_s^\pi \left[\sum_{i=0}^{k-1} c(X_i, \mu_i(X_i)) \right]$$

Thm: the LFP of T_μ is J_μ

Proof: Let $J_0 = 0$ and $\hat{J}$ be the LFP of T_μ.

By Lemma, $(T_\mu^k J_0)(s) = E_s^\mu \left[\sum_{i=0}^{k-1} c(X_i, \mu(X_i)) \right]$.

Hence, $\hat{J}(s) = \lim_{k\to\infty} (T_\mu^k J_0)(s) = E_s^\mu \left[\sum_{i=0}^{\infty} c(X_i, \mu(X_i)) \right] = J_\mu(s)$

Bellman Equation

Let J^* be the LFP of T

Recursion II: Bellman Equation

$$J^*(s) = \min_{a \in A(s)} c(s,a) + \sum_{s'} p(s'|s,a) J^*(s')$$

Bellman Equation

Let J^* be the LFP of T

Recursion II: Bellman Equation

$$J^*(s) = \min_{a \in A(s)} c(s,a) + \sum_{s'} p(s'|s,a) J^*(s')$$

Thm: $J^* \leq J_\pi$ for all π (stationary or not)

Skip Proof

Bellman Equation

Let J^* be the LFP of T

Recursion II: Bellman Equation

$$J^*(s) = \min_{a \in A(s)} c(s,a) + \sum_{s'} p(s'|s,a) J^*(s')$$

Thm: $J^* \leq J_\pi$ for all π (stationary or not)

Proof: show by induction on k that $T^k J_0 \leq T_{\mu_0} \cdots T_{\mu_{k-1}} J_0$

Bellman Equation

Let J^* be the LFP of T

Recursion II: Bellman Equation

$$J^*(s) = \min_{a \in A(s)} c(s,a) + \sum_{s'} p(s'|s,a) J^*(s')$$

Thm: $J^* \leq J_\pi$ for all π (stationary or not)

Proof: show by induction on k that $T^k J_0 \leq T_{\mu_0} \cdots T_{\mu_{k-1}} J_0$

$(T_{\mu_0} \cdots T_{\mu_k} J_0)(s) = c(s, \mu_0(s)) + \sum_{s'} p(s'|s, \mu_0(s))(T_{\mu_1} \cdots T_{\mu_k} J_0)(s')$

Bellman Equation

Let J^* be the LFP of T

Recursion II: Bellman Equation

$$J^*(s) = \min_{a \in A(s)} c(s,a) + \sum_{s'} p(s'|s,a) J^*(s')$$

Thm: $J^* \leq J_\pi$ for all π (stationary or not)

Proof: show by induction on k that $T^k J_0 \leq T_{\mu_0} \cdots T_{\mu_{k-1}} J_0$

$$\begin{aligned}(T_{\mu_0} \cdots T_{\mu_k} J_0)(s) &= c(s,\mu_0(s)) + \textstyle\sum_{s'} p(s'|s,\mu_0(s))(T_{\mu_1} \cdots T_{\mu_k} J_0)(s') \\ &\geq c(s,\mu_0(s)) + \textstyle\sum_{s'} p(s'|s,\mu_0(s))(T^k J_0)(s')\end{aligned}$$

Bellman Equation

Let J^* be the LFP of T

Recursion II: Bellman Equation

$$J^*(s) = \min_{a \in A(s)} c(s,a) + \sum_{s'} p(s'|s,a) J^*(s')$$

Thm: $J^* \leq J_\pi$ for all π (stationary or not)

Proof: show by induction on k that $T^k J_0 \leq T_{\mu_0} \cdots T_{\mu_{k-1}} J_0$

$$\begin{aligned}(T_{\mu_0} \cdots T_{\mu_k} J_0)(s) &= c(s, \mu_0(s)) + \textstyle\sum_{s'} p(s'|s, \mu_0(s))(T_{\mu_1} \cdots T_{\mu_k} J_0)(s') \\ &\geq c(s, \mu_0(s)) + \textstyle\sum_{s'} p(s'|s, \mu_0(s))(T^k J_0)(s') \\ &\geq \min_{a \in A(s)} c(s,a) + \textstyle\sum_{s'} p(s'|s,a)(T^k J_0)(s')\end{aligned}$$

Bellman Equation

Let J^* be the LFP of T

Recursion II: Bellman Equation

$$J^*(s) = \min_{a \in A(s)} c(s,a) + \sum_{s'} p(s'|s,a) J^*(s')$$

Thm: $J^* \leq J_\pi$ for all π (stationary or not)

Proof: show by induction on k that $T^k J_0 \leq T_{\mu_0} \cdots T_{\mu_{k-1}} J_0$

$$\begin{aligned}
(T_{\mu_0} \cdots T_{\mu_k} J_0)(s) &= c(s, \mu_0(s)) + \textstyle\sum_{s'} p(s'|s, \mu_0(s))(T_{\mu_1} \cdots T_{\mu_k} J_0)(s') \\
&\geq c(s, \mu_0(s)) + \textstyle\sum_{s'} p(s'|s, \mu_0(s))(T^k J_0)(s') \\
&\geq \min_{a \in A(s)} c(s,a) + \textstyle\sum_{s'} p(s'|s,a)(T^k J_0)(s') \\
&= (T^{k+1} J_0)(s)
\end{aligned}$$

Greedy Policies

The **greedy** (stationary) policy μ for value function J is

$$\mu(s) = \operatorname*{argmin}_{a \in A(s)} c(s,a) + \textstyle\sum_{s'} p(s'|s,a) J(s')$$

Greedy Policies

The **greedy** (stationary) policy μ for value function J is

$$\mu(s) = \operatorname*{argmin}_{a \in A(s)} c(s,a) + \textstyle\sum_{s'} p(s'|s,a) J(s')$$

Observe

$$\begin{aligned}(T_\mu J)(s) &= c(s,\mu(s)) + \textstyle\sum_{s'} p(s'|s,\mu(s)) J(s') \\ &= \min_a c(s,a) + \textstyle\sum_{s'} p(s'|s,a) J(s) \\ &= (TJ)(s)\end{aligned}$$

Thus, μ is greedy for J iff $T_\mu J = TJ$

Optimal Greedy Policies

Let μ^* be the greedy policy for J^*

Thm (Main): $J^* = J_{\mu^*}$

Cor: μ^* is an optimal policy (and is stationary!)

Skip Proof of Thm

Optimal Greedy Policies

Let μ^* be the greedy policy for J^*

Thm (Main): $J^* = J_{\mu^*}$

Cor: μ^* is an optimal policy (and is stationary!)

Proof of Thm: Let $J_0 = 0$. Show by induction $T_{\mu^*}^k J_0 \leq J^*$:

Optimal Greedy Policies

Let μ^* be the greedy policy for J^*

Thm (Main): $J^* = J_{\mu^*}$

Cor: μ^* is an optimal policy (and is stationary!)

Proof of Thm: Let $J_0 = 0$. Show by induction $T^k_{\mu^*} J_0 \leq J^*$:

$$\begin{aligned}
T^0_{\mu^*} J_0 &= J_0 \leq J^* \\
T^k_{\mu^*} J_0 &= T_{\mu^*} T^{k-1}_{\mu^*} J_0 \\
&\leq T_{\mu^*} J^* \quad \text{(monotonicity of } T_{\mu^*}\text{)} \\
&= T J^* \qquad (\mu^* \text{ is greedy for } J^*) \\
&= J^*
\end{aligned}$$

Suboptimality of Policies

The **suboptimality of π at state s** is $|J_\pi(s) - J^*(s)|$

The **suboptimality of π** is $\|J_\pi - J^*\| = \max_s |J_\pi(s) - J^*(s)|$

More Facts 1/3

Skip More Facts

More Facts 1/3

Thm: if μ is solution, then $T_\mu^k J \rightarrow J_\mu$ for all J with $\|J\| < \infty$

Cor: if μ is solution, T_μ has a **unique** FP J with $\|J\| < \infty$

Skip Proof of Thm

More Facts 1/3

Thm: if μ is solution, then $T_\mu^k J \to J_\mu$ for all J with $\|J\| < \infty$

Cor: if μ is solution, T_μ has a **unique** FP J with $\|J\| < \infty$

Proof of Thm: $|E_s^\mu J(X_k)| \leq P_s^\mu(X_k \notin G)\|J\| = P_s^\mu(T > k)\|J\| \to 0$

More Facts 1/3

Thm: if μ is solution, then $T_\mu^k J \to J_\mu$ for all J with $\|J\| < \infty$

Cor: if μ is solution, T_μ has a **unique** FP J with $\|J\| < \infty$

Proof of Thm: $|E_s^\mu J(X_k)| \leq P_s^\mu(X_k \notin G)\|J\| = P_s^\mu(T > k)\|J\| \to 0$

$$(T_\mu^0 J)(s) = J(s) = (T_\mu^0 J_0)(s) + E_s^\mu J(X_0)$$

More Facts 1/3

Thm: if μ is solution, then $T_\mu^k J \to J_\mu$ for all J with $\|J\| < \infty$

Cor: if μ is solution, T_μ has a **unique** FP J with $\|J\| < \infty$

Proof of Thm: $|E_s^\mu J(X_k)| \leq P_s^\mu(X_k \notin G)\|J\| = P_s^\mu(T > k)\|J\| \to 0$

$$(T_\mu^0 J)(s) = J(s) = (T_\mu^0 J_0)(s) + E_s^\mu J(X_0)$$

$$(T_\mu^k J)(s) = c(s, \mu(s)) + E_s^\mu T_\mu^{k-1} J(X_1)$$

More Facts 1/3

Thm: if μ is solution, then $T^k_\mu J \to J_\mu$ for all J with $\|J\| < \infty$

Cor: if μ is solution, T_μ has a **unique** FP J with $\|J\| < \infty$

Proof of Thm: $|E^\mu_s J(X_k)| \leq P^\mu_s(X_k \notin G)\|J\| = P^\mu_s(T > k)\|J\| \to 0$

$$(T^0_\mu J)(s) = J(s) = (T^0_\mu J_0)(s) + E^\mu_s J(X_0)$$

$$(T^k_\mu J)(s) = c(s, \mu(s)) + E^\mu_s T^{k-1}_\mu J(X_1)$$

$$= c(s, \mu(s)) + E^\mu_s \big[(T^{k-1}_\mu J_0)(X_1) + E^\mu_{X_1} J(X_{k-1})\big]$$

More Facts 1/3

Thm: if μ is solution, then $T_\mu^k J \to J_\mu$ for all J with $\|J\| < \infty$

Cor: if μ is solution, T_μ has a **unique** FP J with $\|J\| < \infty$

Proof of Thm: $|E_s^\mu J(X_k)| \leq P_s^\mu(X_k \notin G)\|J\| = P_s^\mu(T > k)\|J\| \to 0$

$$(T_\mu^0 J)(s) = J(s) = (T_\mu^0 J_0)(s) + E_s^\mu J(X_0)$$

$$\begin{aligned}(T_\mu^k J)(s) &= c(s,\mu(s)) + E_s^\mu T_\mu^{k-1} J(X_1) \\ &= c(s,\mu(s)) + E_s^\mu \big[(T_\mu^{k-1} J_0)(X_1) + E_{X_1}^\mu J(X_{k-1})\big] \\ &= (T_\mu^k J_0)(s) + E_s^\mu \big[E_{X_1}^\mu J(X_{k-1})\big]\end{aligned}$$

More Facts 1/3

Thm: if μ is solution, then $T_\mu^k J \to J_\mu$ for all J with $\|J\| < \infty$

Cor: if μ is solution, T_μ has a **unique** FP J with $\|J\| < \infty$

Proof of Thm: $|E_s^\mu J(X_k)| \leq P_s^\mu(X_k \notin G)\|J\| = P_s^\mu(T > k)\|J\| \to 0$

$$(T_\mu^0 J)(s) = J(s) = (T_\mu^0 J_0)(s) + E_s^\mu J(X_0)$$

$$\begin{aligned}
(T_\mu^k J)(s) &= c(s,\mu(s)) + E_s^\mu T_\mu^{k-1} J(X_1) \\
&= c(s,\mu(s)) + E_s^\mu \left[(T_\mu^{k-1} J_0)(X_1) + E_{X_1}^\mu J(X_{k-1}) \right] \\
&= (T_\mu^k J_0)(s) + E_s^\mu \left[E_{X_1}^\mu J(X_{k-1}) \right] \\
&= (T_\mu^k J_0)(s) + E_s^\mu J(X_k) \to \lim_{k\to\infty} (T_\mu^k J_0)(s) + 0
\end{aligned}$$

More Facts 2/3

Thm: if $T_\mu J \leq J$ for some J such that $\|J\| < \infty$, μ is solution

More Facts 2/3

Thm: if $T_\mu J \leq J$ for some J such that $\|J\| < \infty$, μ is solution

Proof: $J \geq T_\mu J \geq T_\mu^2 J \geq \cdots \geq T_\mu^k J \searrow \hat{J}$. By continuity of T_μ, $\hat{J}$ is FP of T_μ. Therefore, $J_\mu \leq \hat{J} \leq J$. Hence, μ is proper (by Thm)

More Facts 2/3

Thm: if $T_\mu J \leq J$ for some J such that $\|J\| < \infty$, μ is solution

Proof: $J \geq T_\mu J \geq T_\mu^2 J \geq \cdots \geq T_\mu^k J \searrow \hat{J}$. By continuity of T_μ, $\hat{J}$ is FP of T_μ. Therefore, $J_\mu \leq \hat{J} \leq J$. Hence, μ is proper (by Thm)

Thm: if $\exists$ solution, $T^k J \to J^*$ for all J such that $\|J\| < \infty$

Cor: if $\exists$ solution, T has a **unique** FP J such that $\|J\| < \infty$

More Facts 2/3

Thm: if $T_\mu J \leq J$ for some J such that $\|J\| < \infty$, μ is solution

Proof: $J \geq T_\mu J \geq T_\mu^2 J \geq \cdots \geq T_\mu^k J \searrow \hat{J}$. By continuity of T_μ, $\hat{J}$ is FP of T_μ. Therefore, $J_\mu \leq \hat{J} \leq J$. Hence, μ is proper (by Thm)

Thm: if $\exists$ solution, $T^k J \to J^*$ for all J such that $\|J\| < \infty$

Cor: if $\exists$ solution, T has a **unique** FP J such that $\|J\| < \infty$

Proof: Let J, J' be two solutions with $\|J\|, \|J'\| < \infty$. Let μ and μ' be such that $T_\mu J = TJ = J$ and $T_{\mu'} J' = TJ' = J'$

Then, μ and μ' are proper. Hence,

$$J = T^k J \leq T_{\mu'} J \to J'$$
$$J' = T^k J' \leq T_\mu J' \to J$$

Therefore, $J = J'$

More Facts 3/3

Let J_0 be such that $\|J_0\| < \infty$. Define:

- μ_0 greedy for J_0
- μ_1 greedy for J_{μ_0}
- ...
- μ_{k+1} greedy for J_{μ_k}

Thm: μ_k tends to an optimal policy as k tends to ∞

Skip Proof

More Facts 3/3

Let J_0 be such that $\|J_0\| < \infty$. Define:

- μ_0 greedy for J_0
- μ_1 greedy for J_{μ_0}
- $\ldots$
- μ_{k+1} greedy for J_{μ_k}

Thm: μ_k tends to an optimal policy as k tends to ∞

Proof: $J_{\mu_k} = T_{\mu_k} J_{\mu_k} \geq T J_{\mu_k} = T_{\mu_{k+1}} J_{\mu_k}$

More Facts 3/3

Let J_0 be such that $\|J_0\| < \infty$. Define:

- μ_0 greedy for J_0
- μ_1 greedy for J_{μ_0}
- $\ldots$
- μ_{k+1} greedy for J_{μ_k}

Thm: μ_k tends to an optimal policy as k tends to ∞

Proof: $J_{\mu_k} = T_{\mu_k} J_{\mu_k} \geq T J_{\mu_k} = T_{\mu_{k+1}} J_{\mu_k}$

Monotonicity:

More Facts 3/3

Let J_0 be such that $\|J_0\| < \infty$. Define:

- μ_0 greedy for J_0
- μ_1 greedy for J_{μ_0}
- ...
- μ_{k+1} greedy for J_{μ_k}

Thm: μ_k tends to an optimal policy as k tends to ∞

Proof: $J_{\mu_k} = T_{\mu_k} J_{\mu_k} \geq T J_{\mu_k} = T_{\mu_{k+1}} J_{\mu_k}$

Monotonicity: $J_{\mu_k} \geq T_{\mu_{k+1}} J_{\mu_k}$

More Facts 3/3

Let J_0 be such that $\|J_0\| < \infty$. Define:

- μ_0 greedy for J_0
- μ_1 greedy for J_{μ_0}
- $\ldots$
- μ_{k+1} greedy for J_{μ_k}

Thm: μ_k tends to an optimal policy as k tends to ∞

Proof: $J_{\mu_k} = T_{\mu_k} J_{\mu_k} \geq T J_{\mu_k} = T_{\mu_{k+1}} J_{\mu_k}$

Monotonicity: $J_{\mu_k} \geq T_{\mu_{k+1}} J_{\mu_k} \geq T^2_{\mu_{k+1}} J_{\mu_k}$

More Facts 3/3

Let J_0 be such that $\|J_0\| < \infty$. Define:

- μ_0 greedy for J_0
- μ_1 greedy for J_{μ_0}
- $\ldots$
- μ_{k+1} greedy for J_{μ_k}

Thm: μ_k tends to an optimal policy as k tends to ∞

Proof: $J_{\mu_k} = T_{\mu_k} J_{\mu_k} \geq T J_{\mu_k} = T_{\mu_{k+1}} J_{\mu_k}$

Monotonicity: $J_{\mu_k} \geq T_{\mu_{k+1}} J_{\mu_k} \geq T^2_{\mu_{k+1}} J_{\mu_k} \geq \cdots \geq T^m_{\mu_{k+1}} J_{\mu_k} \searrow J_{\mu_{k+1}}$

Hence, $J_{\mu_0} \geq J_{\mu_1} \geq J_{\mu_2} \geq \cdots J_{\mu_k} \geq \cdots$

More Facts 3/3

Let J_0 be such that $\|J_0\| < \infty$. Define:

- μ_0 greedy for J_0
- μ_1 greedy for J_{μ_0}
- $\ldots$
- μ_{k+1} greedy for J_{μ_k}

Thm: μ_k tends to an optimal policy as k tends to ∞

Proof: $J_{\mu_k} = T_{\mu_k} J_{\mu_k} \geq T J_{\mu_k} = T_{\mu_{k+1}} J_{\mu_k}$

Monotonicity: $J_{\mu_k} \geq T_{\mu_{k+1}} J_{\mu_k} \geq T^2_{\mu_{k+1}} J_{\mu_k} \geq \cdots \geq T^m_{\mu_{k+1}} J_{\mu_k} \searrow J_{\mu_{k+1}}$

Hence, $J_{\mu_0} \geq J_{\mu_1} \geq J_{\mu_2} \geq \cdots J_{\mu_k} \geq \cdots$

If μ_k isn't optimal, $\exists s$ with $(TJ_{\mu_k})(s) < (T_{\mu_k} J_{\mu_k})(s) = J_{\mu_k}(s)$

More Facts 3/3

Let J_0 be such that $\|J_0\| < \infty$. Define:

- μ_0 greedy for J_0
- μ_1 greedy for J_{μ_0}
- $\ldots$
- μ_{k+1} greedy for J_{μ_k}

Thm: μ_k tends to an optimal policy as k tends to ∞

Proof: $J_{\mu_k} = T_{\mu_k} J_{\mu_k} \geq T J_{\mu_k} = T_{\mu_{k+1}} J_{\mu_k}$

Monotonicity: $J_{\mu_k} \geq T_{\mu_{k+1}} J_{\mu_k} \geq T^2_{\mu_{k+1}} J_{\mu_k} \geq \cdots \geq T^m_{\mu_{k+1}} J_{\mu_k} \searrow J_{\mu_{k+1}}$

Hence, $J_{\mu_0} \geq J_{\mu_1} \geq J_{\mu_2} \geq \cdots J_{\mu_k} \geq \cdots$

If μ_k isn't optimal, $\exists s$ with $(TJ_{\mu_k})(s) < (T_{\mu_k} J_{\mu_k})(s) = J_{\mu_k}(s)$

$$J_{\mu_{k+1}}(s) \leq (T^m_{\mu_{k+1}} J_{\mu_k})(s) \leq (T_{\mu_{k+1}} J_{\mu_k})(s) = (TJ_{\mu_k})(s) < J_{\mu_k}(s)$$

Summary

- Solutions are not linear plans but functions that map states into actions
- Global solutions vs. solutions for s_{init}
- Cost of solutions is expected cost over trajectories
- Cost function J_μ is LFP of operator T_μ
- There is a stationary policy μ^* that is optimal
- J_{μ^*} satisfies the Bellman equation and is LFP of Bellman operator

Notes

- Probabilistic planning problems is a **subclass** of stochastic shortest-path problems
- There are recent partial results on bounding suboptimality in terms of residual
- If there are no probabilities, have **pure non-deterministic** models
  - **strong cyclic solutions**

Part III

Algorithms

Goals of Part III

- Basic Algorithms
  - Value Iteration and Asynchronous Value Iteration
  - Policy Iteration
  - Linear Programming
- Heuristic Search Algorithms
  - Real-Time Dynamic Programming
  - LAO*
  - Labeled Real-Time Dynamic Programming
  - Others
- Summary and Notes

Value Iteration (VI) 1/2

Computes a sequence of iterates J_k using the Bellman equation as assignment:

$$J_{k+1}(s) = \min_{a \in A(s)} c(s,a) + \sum_{s'} p(s'|s,a) J_k(s')$$

I.e., $J_{k+1} = TJ_k$. The initial iterate is J_0

The iteration stops when the **residual** $\|J_{k+1} - J_k\| < \epsilon$

- Enough to store two vectors: J_k (current) and J_{k+1} (new)
- **Gauss-Seidel**: store one vector (performs update **in place**)

Value Iteration (VI) 2/2

Thm: if there is a solution, $\|J_{k+1} - J_k\| \to 0$ from every initial J_0 with $\|J_0\| < \infty$

Cor: if there is solution, VI terminates in finite time

Value Iteration (VI) 2/2

Thm: if there is a solution, $\|J_{k+1} - J_k\| \to 0$ from every initial J_0 with $\|J_0\| < \infty$

Cor: if there is solution, VI terminates in finite time

Open: upon termination at iterate $k+1$ with residual $< \epsilon$, what is the suboptimality of the greedy policy μ_k for J_k; i.e., $\|J^* - J_{\mu_k}\|$?

Asynchronous Value Iteration

VI doesn't need to update all states in all iterations

Let S_k be the **states updated at iteration** k; i.e.,

$$J_{k+1}(s) = \begin{cases} (TJ_k)(s) & \text{if } s \in S_k \\ J_k(s) & \text{otherwise} \end{cases}$$

Thm: if there is solution and every state is updated **infinitely often**, then $J_k \to J^*$ as $k \to \infty$

Policy Iteration (PI)

Computes a sequence of greedy policies μ_k from initial J_0:

- μ_0 is greedy for J_0
- μ_{k+1} is greedy for J_{μ_k}
- Stops when $J_{\mu_{k+1}} = J_{\mu_k}$

J_{μ_k} is calculated solving the **linear system** $T_{\mu_k} J_{\mu_k} = J_{\mu_k}$

Thm: if there is solution, PI terminates in finite time with an optimal policy

Modified Policy Iteration (MPI)

The computation of J_{μ_k} (**policy evaluation**) is the most time-consuming step in PI. MPI differs from PI in two aspects:

- Policy evaluation is done **iteratively** by computing a sequence $J_{\mu_k}^{m+1} = T_{\mu_k} J_{\mu_k}^{m}$ that converges to J_{μ_k}. This iteration is performed until $\|J_{\mu_k}^{m+1} - J_{\mu_k}^{m}\| < \delta$
- MPI is **stopped** when $\|J_{\mu_{k+1}} - J_{\mu_k}\| < \epsilon$

Linear Programming (LP)

Optimal value function J^* computed as the solution of the LP:

$$\begin{aligned} & \max \sum_s x_s \\ \text{s.t.} \quad & c(s,a) + \sum_s p(s'|s,a) x_{s'} \ \geq \ x_s \quad \forall s \in S, a \in A(s) \\ & x_s \geq 0 \quad \forall s \in S \end{aligned}$$

Thm: if there is solution, LP has bounded solution $\{x_s\}_{s \in S}$ and $J^*(s) = x_s$ for all $s \in S$

Linear Programming (LP)

Optimal value function J^* computed as the solution of the LP:

$$\max \sum_s x_s$$

$$\text{s.t.} \quad c(s,a) + \sum_s p(s'|s,a) x_{s'} \ \geq \ x_s \quad \forall s \in S, a \in A(s)$$

$$x_s \geq 0 \quad \forall s \in S$$

Thm: if there is solution, LP has bounded solution $\{x_s\}_{s \in S}$ and $J^*(s) = x_s$ for all $s \in S$

In practice, VI is faster than PI, MPI and LP

Discussion

Complete methods compute entire solutions (policies) that work for all states. In probabilistic planning, we want solutions for s_{init}

Worse, the problem may have solution for s_{init} and not have entire solution (e.g., when there are **avoidable dead-end** states). In this case, previous methods do not work

Search-based methods designed to compute solutions for s_{init}

Real-Time Dynamic Programming (RTDP)

Let H be empty hash table with entries $H(s)$ initialized to $h(s)$ as needed
repeat
  Set $s := s_{init}$
  while $s \notin G$ **do**
    For each action $a \in A(s)$, set $Q(s,a) := c(s,a) + \sum_{s' \in S} p(s'|s,a)H(s')$
    Select best action $\mathbf{a} := \text{argmin}_{a \in A(s)} Q(s,a)$
    Update value $H(s) := Q(s,\mathbf{a})$
    Sample next state s' with probability $p(s'|s,\mathbf{a})$ and set $s := s'$
  end while
until some termination condition

- **Online algorithm** that interleaves planning/execution
- Performs multiple **trials**. At each, actions chosen with **one-step lookahead** using current value function
- Converges to optimal policy under certain conditions
- Can be made into an offline algorithm
- Generalizes Korf's **Learning Real-Time A*** (LRTA*)

Properties of Heuristics

Heuristic $h : S \rightarrow \mathbb{R}^+$ is **admissible** if $h \leq J^*$

Heuristic $h : S \rightarrow \mathbb{R}^+$ is **consistent** if $h \leq Th$

Lemma: if h is consistent, h is admissible

Properties of Heuristics

Heuristic $h : S \to \mathbb{R}^+$ is **admissible** if $h \leq J^*$

Heuristic $h : S \to \mathbb{R}^+$ is **consistent** if $h \leq Th$

Lemma: if h is consistent, h is admissible

Proof: (by monotonicity of T) $h \leq Th \leq T^2h \leq \cdots \leq J^*$

Properties of Heuristics

Heuristic $h : S \to \mathbb{R}^+$ is **admissible** if $h \leq J^*$

Heuristic $h : S \to \mathbb{R}^+$ is **consistent** if $h \leq Th$

Lemma: if h is consistent, h is admissible

Proof: (by monotonicity of T) $h \leq Th \leq T^2h \leq \cdots \leq J^*$

Lemma: let h be consistent (admissible) and $h' = h$ except $h'(s') = (Th)(s')$. Then, h' is consistent (admissible)

Properties of Heuristics

Heuristic $h : S \to \mathbb{R}^+$ is **admissible** if $h \leq J^*$

Heuristic $h : S \to \mathbb{R}^+$ is **consistent** if $h \leq Th$

Lemma: if h is consistent, h is admissible

Proof: (by monotonicity of T) $h \leq Th \leq T^2 h \leq \cdots \leq J^*$

Lemma: let h be consistent (admissible) and $h' = h$ except $h'(s') = (Th)(s')$. Then, h' is consistent (admissible)

Proof: $h(s') \leq (Th)(s') = h'(s')$ and $h(s) = h'(s)$. Hence, $h \leq h'$

$$h'(s) = h(s) \leq (Th)(s) \leq (Th')(s)$$

$$h'(s') = (Th)(s') \leq (Th')(s')$$

Properties of Heuristics

Heuristic $h : S \to \mathbb{R}^+$ is **admissible** if $h \leq J^*$

Heuristic $h : S \to \mathbb{R}^+$ is **consistent** if $h \leq Th$

Lemma: if h is consistent, h is admissible

Proof: (by monotonicity of T) $h \leq Th \leq T^2 h \leq \cdots \leq J^*$

Lemma: let h be consistent (admissible) and $h' = h$ except $h'(s') = (Th)(s')$. Then, h' is consistent (admissible)

Proof: $h(s') \leq (Th)(s') = h'(s')$ and $h(s) = h'(s)$. Hence, $h \leq h'$

$$h'(s) = h(s) \leq (Th)(s) \leq (Th')(s)$$

$$h'(s') = (Th)(s') \leq (Th')(s')$$

The constant-zero heuristic is **admissible and consistent**

Subproblems

Subproblems:

Let $M = \langle S, A, G, s_{init}, p(\cdot), c(\cdot) \rangle$ be a probabilistic planning problem

A subproblem of M is $M' = \langle S', A', G', s'_{init}, p'(\cdot), c'(\cdot) \rangle$ such that

- $S' \subseteq S$
- $A'(s) \subseteq A(s)$ for all $s \in S'$
- $G' \subseteq G$
- $s'_{init} = s_{init}$
- for all $s \in S'$ and $a \in A'(s)$, if $p(s'|s, a) > 0$ then $s' \in S'$
- $p'(s'|s, a) = p(s'|s, a)$ for all $s \in S'$ and $a \in A'(s)$
- $c'(s, a) = c(s, a)$ for all $s \in S'$ and $a \in A'(s)$

Thm: if M' is subproblem of M, $J^*_M(s) \leq J_{M'}(s)$ for $s \in S'$

Convergence of RTDP

Thm: if $\exists$ solution for reachable states from s_{init}, and h is admissible, RTDP converges to optimal policy for s_{init} w.p. 1

Skip Proof

Convergence of RTDP

Thm: if $\exists$ solution for reachable states from s_{init}, and h is admissible, RTDP converges to optimal policy for s_{init} w.p. 1

Proof: Let J_k be the value function at time k; e.g., $J_0 = h$

Convergence of RTDP

Thm: if $\exists$ solution for reachable states from s_{init}, and h is admissible, RTDP converges to optimal policy for s_{init} w.p. 1

Proof: Let J_k be the value function at time k; e.g., $J_0 = h$

By Lemma, J_k is admissible and bounded; i.e. $J_k \leq J^*$ and $\|J^*\| < \infty$

Convergence of RTDP

Thm: if $\exists$ solution for reachable states from s_{init}, and h is admissible, RTDP converges to optimal policy for s_{init} w.p. 1

Proof: Let J_k be the value function at time k; e.g., $J_0 = h$

By Lemma, J_k is admissible and bounded; i.e. $J_k \leq J^*$ and $\|J^*\| < \infty$

Let $\Omega = \{\langle s_0, s_1, \ldots \rangle : s_0 = s_{init}\}$ be the trajectories RTDP can generate, and $\mathbb{P}$ the measure on Ω defined by the **random choices** in RTDP

Convergence of RTDP

Thm: if $\exists$ solution for reachable states from s_{init}, and h is admissible, RTDP converges to optimal policy for s_{init} w.p. 1

Proof: Let J_k be the value function at time k; e.g., $J_0 = h$

By Lemma, J_k is admissible and bounded; i.e. $J_k \leq J^*$ and $\|J^*\| < \infty$

Let $\Omega = \{\langle s_0, s_1, \ldots \rangle : s_0 = s_{init}\}$ be the trajectories RTDP can generate, and $\mathbb{P}$ the measure on Ω defined by the **random choices** in RTDP

For trajectory $\tau \in \Omega$, define:

- set S_τ of **recurrent** states in τ
- sets $A_\tau(s)$ of recurrent actions in τ chosen at $s \in S_\tau$

Convergence of RTDP

Thm: if $\exists$ solution for reachable states from s_{init}, and h is admissible, RTDP converges to optimal policy for s_{init} w.p. 1

Proof: Let J_k be the value function at time k; e.g., $J_0 = h$

By Lemma, J_k is admissible and bounded; i.e. $J_k \leq J^*$ and $\|J^*\| < \infty$

Let $\Omega = \{\langle s_0, s_1, \ldots \rangle : s_0 = s_{init}\}$ be the trajectories RTDP can generate, and $\mathbb{P}$ the measure on Ω defined by the **random choices** in RTDP

For trajectory $\tau \in \Omega$, define:

- set S_τ of **recurrent** states in τ
- sets $A_\tau(s)$ of recurrent actions in τ chosen at $s \in S_\tau$

For $s, a, p(s'|s,a) > 0$, let $B(s,a,s') = \{\tau : s \in S_\tau, a \in A_\tau(s), s' \notin S_\tau\}$

Convergence of RTDP

Thm: if $\exists$ solution for reachable states from s_{init}, and h is admissible, RTDP converges to optimal policy for s_{init} w.p. 1

Proof: Let J_k be the value function at time k; e.g., $J_0 = h$

By Lemma, J_k is admissible and bounded; i.e. $J_k \leq J^*$ and $\|J^*\| < \infty$

Let $\Omega = \{\langle s_0, s_1, \ldots \rangle : s_0 = s_{init}\}$ be the trajectories RTDP can generate, and $\mathbb{P}$ the measure on Ω defined by the **random choices** in RTDP

For trajectory $\tau \in \Omega$, define:

- set S_τ of **recurrent** states in τ
- sets $A_\tau(s)$ of recurrent actions in τ chosen at $s \in S_\tau$

For $s, a, p(s'|s,a) > 0$, let $B(s,a,s') = \{\tau : s \in S_\tau, a \in A_\tau(s), s' \notin S_\tau\}$

$\mathbb{P}(B(s,a,s')) = 0 \implies \mathbb{P}(\cup_{s,a,s'} B(s,a,s')) = 0$

Convergence of RTDP

Thm: if $\exists$ solution for reachable states from s_{init}, and h is admissible, RTDP converges to optimal policy for s_{init} w.p. 1

Proof: Let J_k be the value function at time k; e.g., $J_0 = h$

By Lemma, J_k is admissible and bounded; i.e. $J_k \leq J^*$ and $\|J^*\| < \infty$

Let $\Omega = \{\langle s_0, s_1, \ldots \rangle : s_0 = s_{init}\}$ be the trajectories RTDP can generate, and $\mathbb{P}$ the measure on Ω defined by the **random choices** in RTDP

For trajectory $\tau \in \Omega$, define:

- set S_τ of **recurrent** states in τ
- sets $A_\tau(s)$ of recurrent actions in τ chosen at $s \in S_\tau$

For $s, a, p(s'|s,a) > 0$, let $B(s,a,s') = \{\tau : s \in S_\tau, a \in A_\tau(s), s' \notin S_\tau\}$

$\mathbb{P}(B(s,a,s')) = 0 \implies \mathbb{P}(\cup_{s,a,s'} B(s,a,s')) = 0$

Hence, S_τ and A_τ define a **random subproblem** M_τ

Convergence of RTDP (Cont'd)

With $\mathbb{P}$-probability 1:

- After finite time, RTDP performs Asynchronous VI on M_τ and thus converges to the value function J_{M_τ} on S_τ

Convergence of RTDP (Cont'd)

With $\mathbb{P}$-probability 1:

- After finite time, RTDP performs Asynchronous VI on M_τ and thus converges to the value function J_{M_τ} on S_τ
- $J^*(s) \leq J_{M_\tau}(s)$ for all $s \in S_\tau$

Convergence of RTDP (Cont'd)

With $\mathbb{P}$-probability 1:

- After finite time, RTDP performs Asynchronous VI on M_τ and thus converges to the value function J_{M_τ} on S_τ
- $J^*(s) \leq J_{M_\tau}(s)$ for all $s \in S_\tau$
- On the other hand, $J_k \leq J^*$ for all k

Convergence of RTDP (Cont'd)

With $\mathbb{P}$-probability 1:

- After finite time, RTDP performs Asynchronous VI on M_τ and thus converges to the value function J_{M_τ} on S_τ
- $J^*(s) \leq J_{M_\tau}(s)$ for all $s \in S_\tau$
- On the other hand, $J_k \leq J^*$ for all k

Therefore, RTDP converges to $J_{M_\tau}(s) = J^*(s)$ for all $s \in S_\tau$

Convergence of RTDP (Cont'd)

With $\mathbb{P}$-probability 1:

- After finite time, RTDP performs Asynchronous VI on M_τ and thus converges to the value function J_{M_τ} on S_τ
- $J^*(s) \leq J_{M_\tau}(s)$ for all $s \in S_\tau$
- On the other hand, $J_k \leq J^*$ for all k

Therefore, RTDP converges to $J_{M_\tau}(s) = J^*(s)$ for all $s \in S_\tau$

It remains to show that s_{init} is recurrent; i.e., $s_{init} \in S_\tau$ w.p. 1

Convergence of RTDP (Cont'd)

With $\mathbb{P}$-probability 1:

- After finite time, RTDP performs Asynchronous VI on M_τ and thus converges to the value function J_{M_τ} on S_τ
- $J^*(s) \leq J_{M_\tau}(s)$ for all $s \in S_\tau$
- On the other hand, $J_k \leq J^*$ for all k

Therefore, RTDP converges to $J_{M_\tau}(s) = J^*(s)$ for all $s \in S_\tau$

It remains to show that s_{init} is recurrent; i.e., $s_{init} \in S_\tau$ w.p. 1

If $s_{init} \notin S_\tau$, then S_τ contains no goal state

Convergence of RTDP (Cont'd)

With $\mathbb{P}$-probability 1:

- After finite time, RTDP performs Asynchronous VI on M_τ and thus converges to the value function J_{M_τ} on S_τ
- $J^*(s) \leq J_{M_\tau}(s)$ for all $s \in S_\tau$
- On the other hand, $J_k \leq J^*$ for all k

Therefore, RTDP converges to $J_{M_\tau}(s) = J^*(s)$ for all $s \in S_\tau$

It remains to show that s_{init} is recurrent; i.e., $s_{init} \in S_\tau$ w.p. 1

If $s_{init} \notin S_\tau$, then S_τ contains no goal state

Therefore, all costs in M_τ are positive (there are no absorbing states). Hence, $J_{M_\tau}(s) = \infty$ for $s \in S_\tau$ which is impossible since $\|J^*\| < \infty$

AND/OR Graphs

An AND/OR graph is a **rooted digraph** made of AND nodes and OR nodes

An OR node represents the **choice** of an action at the state

An AND node represents (multiple) **outcomes** of an action

if deterministic actions, the AND/OR graph is a digraph

Solution for AND/OR Graphs

A solution to an AND/OR graph is a **subgraph** that satisfies:

- the root node (for s_{init}) belongs to the solution
- for every OR node in the solution, exactly one of its branches belongs to the solution
- for every AND node in the solution, all of its branches belong to the solution

Complete if every maximal directed path ends in a goal

Partial if any directed path ends at an open (unexpanded) node

AO* = A* for AND/OR Graphs

Best First: iteratively, expands nodes on the fringe of best partial solution graph until solution is complete

Optimal because cost of best partial solution graph is lower bound of any complete solution

Best partial solution determined **greedily** by choosing, for each OR node, the action with **best (expected) value**

AO* solves the DP recursion in **acyclic spaces** by:

- **Expansion:** expands one or more nodes on the fringe of best partial solution
- **Cost Revision:** propagates the new values on the fringe using **backward induction**

LAO*

LAO* generalizes AO* for **AND/OR graphs with cycles**

LAO*

LAO* generalizes AO* for **AND/OR graphs with cycles**

Maintains the expansion step of AO* but changes the cost-revision step from backward induction to **policy evaluation** of the partial solution

LAO*

LAO* generalizes AO* for **AND/OR graphs with cycles**

Maintains the expansion step of AO* but changes the cost-revision step from backward induction to **policy evaluation** of the partial solution

Improved LAO* (ILAO*):

- expands all open nodes on the fringe of current solution
- performs just **one backup** for each node in current solution

LAO*

LAO* generalizes AO* for **AND/OR graphs with cycles**

Maintains the expansion step of AO* but changes the cost-revision step from backward induction to **policy evaluation** of the partial solution

Improved LAO* (ILAO*):

- expands all open nodes on the fringe of current solution
- performs just **one backup** for each node in current solution

As a result, current partial solution is not **guaranteed** to be a best partial solution

Hence, stopping criteria is **strengthened to ensure optimality**

Improved LAO*

Explicit graph initially consists of the start state s_{init}
repeat
  Depth-first traversal of states in current best (partial) solution graph
  foreach visited state s in postorder traversal **do**
    if state s isn't expanded **then**
      Expand s by generating each successor s' and initializing $H(s')$ to $h(s')$
    end if
    Set $H(s) := \min_{a \in A(s)} c(s,a) + \sum_{s' \in S} p(s'|s,a)H(s')$ and mark best action
  end foreach
until best solution graph hos no unexpanded tips and residual $< \epsilon$

The expansion and cost-revision steps of ILAO* performed in the **same depth-first traversal** of the partial solution graph

Stopping criteria extended with a **test on residual**

Thm: if there is solution for s_{init} and h is consistent, LAO* and ILAO* terminate with solution for s_{init} and residual $< \epsilon$

Faster Convergence

ILAO* converges much faster than RTDP because

- performs **systematic exploration** of the state space rather than **stochastic exploration**
- has an **explicit convergence test**

Both ideas can be incorporated into RTDP

Labeling States

RTDP keeps visiting reachable states even when the value function has **converged** over them

Updates over “solved states” are **wasteful** as the value function doesn’t change

Hence, it makes sense to **detect solved states** and to avoid performing updates on them

Solved States

A state s is said to be **solved** for J when s and all states reachable from s using π_J have residual $< \epsilon$

If the **solution graph contains cycles**, labeling states as 'solved' **cannot** be done by backward induction

However, the solution graph can be decomposed into strongly-connected components (SCCs) that make up an **acyclic graph that can be labeled**

Detecting Solved States

A depth-first traversal from s that chooses actions with π_J can be used to test if s is solved:

- **backtrack** at solved states returning **true**
- **backtrack** at states with residual $\geq \epsilon$ returning **false**

If updates are performed at states with residual $\geq \epsilon$ and their ancestors, the traversal either

- detects a solved state or
- performs at least one update changing the value

This algorithm is called **CheckSolved**(s)

Labeled RTDP (LRTDP)

Goal states are marked as solved and trials modified to:

- **terminate** at solved states rather than goal states
- at termination, call **CheckSolved** on all states in the trial (in reverse order) until it returns **false**
- **terminate trials** when s_{init} is labeled as solved

Labeled RTDP (LRTDP)

Goal states are marked as solved and trials modified to:

- **terminate** at solved states rather than goal states
- at termination, call **CheckSolved** on all states in the trial (in reverse order) until it returns **false**
- **terminate trials** when s_{init} is labeled as solved

Hence, LRTDP achieves the following:

- **crisp** termination condition
- final function has residual $< \epsilon$ on states reachable from s_{init}
- it doesn't perform updates over converged states
- the search is still stochastic yet is **"more systematic"**

Labeled RTDP (LRTDP)

Goal states are marked as solved and trials modified to:

- **terminate** at solved states rather than goal states
- at termination, call **CheckSolved** on all states in the trial (in reverse order) until it returns **false**
- **terminate trials** when s_{init} is labeled as solved

Hence, LRTDP achieves the following:

- **crisp** termination condition
- final function has residual $< \epsilon$ on states reachable from s_{init}
- it doesn't perform updates over converged states
- the search is still stochastic yet is **"more systematic"**

Thm: if there is solution for all reachable states from s_{init}, and h is consistent, LRTDP terminates with an optimal solution for s_{init} in a number of trials bounded by $\epsilon^{-1} \sum_s J^*(s) - h(s)$

Non-Admissible Heuristics

Suppose there is a solution for s_{init}

Consider **non-admissible** heuristic h

Then, LAO* and LRTDP terminate with **solution** for s_{init}

Heuristic Dynamic Programming (HDP)

Tarjan's algorithm for computing SCCs is a depth-first traversal that computes the SCCs and their acyclic structure

It can be modified to:

- **backtrack** on solved states
- **expand and update** the value of **non-goal tip nodes**
- **update** the value of states with residual $\geq \epsilon$
- **update** the value of **ancestors** of updated nodes
- when detecting an SCC of nodes with residual $< \epsilon$, **label all nodes** in the SCC as **solved**

Modified Tarjan's algorithm can find optimal solutions:

while s_{init} isn't solved **do TarjanSCC**(s_{init})

Other Algorithms

Bounds: admissible heuristics are LBs. With UBs, we can:

- use difference of bounds to bound suboptimality
- use difference of bounds to focus the search

Algorithms that use both bounds are BRTDP, FRTDP, ...

AND/OR Graphs: used to model a variety of problems. The **LDFS** algorithm is a unified algorithm for AND/OR graphs that is based of depth-first search and DP

Symbolic Search: many variants of above algorithms as well as others that implement search in symbolic representations and **factored MDPs**

Summary

- Explicit algorithms such as VI and PI work well for small problems
- Explicit algorithms compute (entire) optimal solutions
- Search algorithms such as LAO* and LRTDP compute solutions for s_{init}
  - if heuristic is admissible, both compute optimal solutions
  - if heuristic is non-admissible, both compute solutions
  - number of updates depends on quality of heuristic
- There are other search algorithms

Part IV

Language

Goals of Part IV

- Representation Languages
  - Flat
  - Factored
- Probabilistic PDDL (PPDDL)
- Simple-PPDDL
- Summary and Notes

Flat Representation

A flat representation consists of explicit:

- enumeration of state space and actions
- boolean vectors for applicability for each action
- transition matrices for each action
- cost vectors for each action

A model with n states and m actions has a flat representation of size $O(n^2m)$ bits

Impractical for problems with many states (millions)

Factored Representation

A factored representation can express a "large" problem with "few" bits

If problem has n states and m actions, a factored representation needs $O(poly(\log nm))$ or $O(poly(m \log n))$ bits

Many variants:

- propositional languages
- dynamic Bayesian networks
- state abstractions
- first-order and relational representations (even more compact!)
- ...

Probabilistic PDDL (PPDDL) 1/2

Extension of PDDL with probabilistic effects and rewards

By Younes and Littman for IPC-04 and extended for IPC-06

New **probabilistic effect** of the form

$$\texttt{(probabilistic <eff}_1\texttt{> <p}_1\texttt{> ... <eff}_n\texttt{> <p}_n\texttt{>)}$$

Effect is well defined iff $0 < p_i \leq 1$ and $\sum_i p_i \leq 1$

Probabilistic PDDL (PPDDL) 1/2

Extension of PDDL with probabilistic effects and rewards

By Younes and Littman for IPC-04 and extended for IPC-06

New **probabilistic effect** of the form

$$\texttt{(probabilistic <eff}_1\texttt{> <p}_1\texttt{> ... <eff}_n\texttt{> <p}_n\texttt{>)}$$

Effect is well defined iff $0 < p_i \leq 1$ and $\sum_i p_i \leq 1$

The effect of the statement is to apply $\texttt{eff}_i$ with probability p_i, or apply null effect with probability $1 - \sum_i p_i$

Example: Blocksworld

```
(:action put-on-block
  :parameters (?b1 ?b2 - block)
  :precondition (and (holding ?b1) (clear ?b2))
  :effect (probabilistic
     3/4 (and (on ?b1 ?b2) (emptyhand) (clear ?b1) (not (holding ?b1)) (not (clear ?b2)))
     1/4 (and (on-table ?b1) (emptyhand) (clear ?b1) (not (holding ?b1))))
)

(:action pick-tower
  :parameters (?b1 ?b2 - block)
  :precondition (and (emptyhand) (on ?b1 ?b2))
  :effect (probabilistic
     1/10 (and (holding ?b1) (clear ?b2) (not (emptyhand)) (not (on ?b1 ?b2))))
)

(:action put-tower-on-block
  :parameters (?b1 ?b2 - block)
  :precondition (and (holding ?b1) (clear ?b2))
  :effect (probabilistic
     1/10 (and (on ?b1 ?b2) (emptyhand) (not (holding ?b1)) (not (clear ?b2)))
     9/10 (and (on-table ?b1) (emptyhand) (not (holding ?b1))))
)
```

Example: Boxworld

```
(:action drive-truck
  :parameters (?t - truck ?src - city ?dst - city)
  :precondition (and (truck-at-city ?t ?src) (can-drive ?src ?dst))
  :effect
    (and (not (truck-at-city ?t ?src))
         (probabilistic
           2/10 (forall (?wrongdst1 - city)
                  (when (wrong-drive1 ?src ?wrongdst1)
                    (forall (?wrongdst2 - city)
                      (when (wrong-drive2 ?src ?wrongdst2)
                        (forall (?wrongdst3 - city)
                          (when (wrong-drive3 ?src ?wrongdst3)
                            (probabilistic
                              1/3 (truck-at-city ?t ?wrongdst1)
                              1/3 (truck-at-city ?t ?wrongdst2)
                              1/3 (truck-at-city ?t ?wrongdst3)
                            )))))))
           8/10 (truck-at-city ?t ?dst)
         )
    )
)
```

Probabilistic PDDL (PPDDL) 2/2

PDDL has **existential** quantification, **disjunctive** conditions, **unbounded nesting** of conditional effects

PPDDL has **unbounded nesting** of conditional and probabilistic effects

Calculating the overall effect is complex

Fragment Simple-PPDDL

Better to focus on **fragment** obtained by **forbidding**:

- existential quantification
- disjunctive conditions
- nested conditional effects
- probabilistic effects inside conditional effects

Fragment called (in this talk) Simple-PPDDL

Fragment Simple-PPDDL

Better to focus on **fragment** obtained by **forbidding**:

- existential quantification
- disjunctive conditions
- nested conditional effects
- probabilistic effects inside conditional effects

Fragment called (in this talk) Simple-PPDDL

Simple-PPDDL is general and permits heuristic for classical planning to be lifted for probabilistic planning

Example: Drive

```
(:action wait_on_light
  :parameters (?x - coord ?y - coord)
  :precondition (and (light_color red) (at ?x ?y))
  :effect
    (and (probabilistic 1/100 (not (alive)))  ; <-- DEAD END!

         (probabilistic 1/2 (when (light_delay ?x ?y quick)
                                  (and (not (light_color red))(light_color green))))

         (probabilistic 1/5 (when (light_delay ?x ?y normal)
                                  (and (not (light_color red))(light_color green))))

         (probabilistic 1/10 (when (light_delay ?x ?y slow)
                                   (and (not (light_color red))(light_color green))))
    )
)
```

Example: Drive

```
(:action look_at_light
  :parameters (?x - coord ?y - coord)
  :precondition (and (light_color unknown) (at ?x ?y))
  :effect
    (and (probabilistic
           9/10 (when (and (heading north) (light_preference ?x ?y north_south))
                  (and (not (light_color unknown))(light_color green)))
           1/10 (when (and (heading north) (light_preference ?x ?y north_south))
                  (and (not (light_color unknown))(light_color red))))

         (probabilistic
           9/10 (when (and (heading south) (light_preference ?x ?y north_south))
                  (and (not (light_color unknown))(light_color green)))
           1/10 (when (and (heading south) (light_preference ?x ?y north_south))
                  (and (not (light_color unknown))(light_color red))))

         (probabilistic
           1/10 (when (and (heading east) (light_preference ?x ?y north_south))
                  (and (not (light_color unknown))(light_color green)))
           9/10 (when (and (heading east) (light_preference ?x ?y north_south))
                  (and (not (light_color unknown))(light_color red))))

         ... ; six more probabilistic effects!

    )
)
```

Summary

- Factored representation specifies a problem with few bits
- PPDDL, Simple-PPDDL and other variants widely used
- **Challenge for algorithms and heuristics!**

Part V

Heuristics

Goals of Part V

- Properties of Heuristics
- How to Obtain Heuristics?
  - Determinization
  - Abstractions
  - Linear Programming
- Representation Languages
- Lifting Heuristics
- Summary and Notes

Recap: Properties of Heuristics

Heuristic $h : S \rightarrow \mathbb{R}^+$ is **admissible** if $h \leq J^*$

Heuristic $h : S \rightarrow \mathbb{R}^+$ is **consistent** if $h \leq Th$

Lemma: if h is consistent, h is admissible

Search-based algorithms compute:

- Optimal solution for s_{init} is heuristic is admissible
- Solution for s_{init} for any heuristic

How to Obtain Admissible Heuristics?

Relax problem → Solve optimally → Admissible heuristic

How to Obtain Admissible Heuristics?

Relax problem $\rightarrow$ Solve optimally $\rightarrow$ Admissible heuristic

How to relax?

- Remove non-determinism
- State abstraction (?)

How to Obtain Admissible Heuristics?

Relax problem $\rightarrow$ Solve optimally $\rightarrow$ Admissible heuristic

How to relax?

- Remove non-determinism
- State abstraction (?)

How to solve relaxation?

- Solver
- Use search with admissible heuristic
- Substitute with admissible heuristic for relaxation

Determinization: Min-Min Heuristic

Determinization **obtained** by transforming Bellman equation

$$J^*(s) = \min_{a \in A(s)} c(s,a) + \sum_{s' \in s} p(s'|s,a) J^*(s')$$

into

$$J^*_{min}(s) = \min_{a \in A(s)} c(s,a) + \min\{J^*_{min}(s') : p(s'|s,a) > 0\}$$

Determinization: Min-Min Heuristic

Determinization **obtained** by transforming Bellman equation

$$J^*(s) = \min_{a \in A(s)} c(s,a) + \sum_{s' \in s} p(s'|s,a) J^*(s')$$

into

$$J^*_{min}(s) = \min_{a \in A(s)} c(s,a) + \min\{J^*_{min}(s') : p(s'|s,a) > 0\}$$

Obs: Bellman equation for **deterministic** problem

Thm: $J^*_{min}(s)$ is consistent and thus $J^*_{min}(s) \leq J^*(s)$

Determinization: Min-Min Heuristic

Determinization **obtained** by transforming Bellman equation

$$J^*(s) = \min_{a \in A(s)} c(s,a) + \sum_{s' \in s} p(s'|s,a) J^*(s')$$

into

$$J^*_{min}(s) = \min_{a \in A(s)} c(s,a) + \min\{J^*_{min}(s') : p(s'|s,a) > 0\}$$

Obs: Bellman equation for **deterministic** problem

Thm: $J^*_{min}(s)$ is consistent and thus $J^*_{min}(s) \leq J^*(s)$

Solve with standard search algorithm, or use **admissible** estimate for J^*_{min}

Abstractions

Abstraction of problem P with space S is problem P' with space S' together with **abstraction function** $\alpha : S \to S'$

Abstractions

Abstraction of problem P with space S is problem P' with space S' together with **abstraction function** $\alpha : S \to S'$

Interested in "small" abstractions; i.e., $|S'| \ll |S|$

Abstractions

Abstraction of problem P with space S is problem P' with space S' together with **abstraction function** $\alpha : S \rightarrow S'$

Interested in "small" abstractions; i.e., $|S'| \ll |S|$

Abstraction is **admissible** if $J^*_{P'}(\alpha(s)) \leq J^*_P(s)$

Abstraction is **bounded** if $J^*_{P'}(\alpha(s)) = \infty \implies J^*_P(s) = \infty$

Abstractions

Abstraction of problem P with space S is problem P' with space S' together with **abstraction function** $\alpha : S \to S'$

Interested in "small" abstractions; i.e., $|S'| \ll |S|$

Abstraction is **admissible** if $J^*_{P'}(\alpha(s)) \leq J^*_P(s)$

Abstraction is **bounded** if $J^*_{P'}(\alpha(s)) = \infty \implies J^*_P(s) = \infty$

Open: how to compute a small abstraction that is admissible?

Open: how to compute a small abstraction that is bounded?

Heuristics + Representation Language

Question: how to compute a heuristic for factored problem?

Heuristics + Representation Language

Question: how to compute a heuristic for factored problem?

Const.: cannot convert factored problem into flat model

Heuristics + Representation Language

Question: how to compute a heuristic for factored problem?

Const.: cannot convert factored problem into flat model

Answer: need to compute heuristic at **representation level**

Determinization of Simple-PPDDL = PDDL

Simple-PPDDL: **un-nested top-level** probabilistic effects

Idea: translate operator with single probabilistic effect with n effects, into n PDDL operators

A Simple-PPDDL operator with m probabilistic effects, generates $2^{O(m)}$ PDDL operators

Convert Simple-PPDDL problems into PDDL (**relaxation**)

Determinization of Simple-PPDDL = PDDL

Simple-PPDDL: **un-nested top-level** probabilistic effects

Idea: translate operator with single probabilistic effect with n effects, into n PDDL operators

A Simple-PPDDL operator with m probabilistic effects, generates $2^{O(m)}$ PDDL operators

Convert Simple-PPDDL problems into PDDL (**relaxation**)

Translation exponential in number of probabilistic effecs

Lifting Classical Planning Heuristics

Let P be a Simple-PPDDL problem and P' its PDDL translation

We have:

- $J^*_{P'} \leq J^*_P$
- admissible heuristic for P' is admissible heuristic for P

Admissible heuristics for classical planning can be **lifted** for probabilistic planning for Simple-PPDDL

Lifting Classical Planning Heuristics

Let P be a Simple-PPDDL problem and P' its PDDL translation

We have:

- $J^*_{P'} \leq J^*_P$
- admissible heuristic for P' is admissible heuristic for P

Admissible heuristics for classical planning can be **lifted** for probabilistic planning for Simple-PPDDL

Therefore, can use admissible LM-Cut, Merge-and-Shrink, $h^m(s)$, etc. and non-admissible h_{FF}, h_{LAMA}

Summary

- Not much known about heuristics for probabilistic planning
- There are (search) algorithms but cannot be **exploited**
- Heuristics to be **effective** must be computed at representation level like done in classical planning
- Heuristics for classical planning can be **lifted** for probabilistic planning through **determinization**

Summary

- Not much known about heuristics for probabilistic planning
- There are (search) algorithms but cannot be **exploited**
- Heuristics to be **effective** must be computed at representation level like done in classical planning
- Heuristics for classical planning can be **lifted** for probabilistic planning through **determinization**

Lots of things to be done about heuristics!

Part VI

Variants and Extensions

Variants and Extensions

- Non-deterministic problems
- MDPs and Stochastic Shortest-Path problems
- Roll-out policies
- Partially Observable MDPs (POMDPs)
- Decentralized POMDPs (DEC-POMDPs)
- Partially Observable Stochastic Games (POSGs)