Procesos controlados de Markov

José Antonio Camarena Ibarrola

Procesos controlados de Markov

- Son una clase de procesos que tratan con el problema de toma de decisiones bajo incertidumbre
- Esta clase incluye a los Procesos de decisión de Markov (MDP); a los Procesos de decisión semi-markovianos (SMDP); y a los procesos de decisión de Markov parcialmente observables (POMDP)
- Son modelos matemáticos utilizados para encontrar estrategias óptimas que un tomador de decisiones debe seguir cuando debe tomar una secuencia de decisiones en el tiempo y cuyos resultados son inciertos.

Programación Dinámica

- Técnica de optimización para problemas de decisión multi-etapa, en los cuales en cada etapa se debe optimizar una variable.
- Un algoritmo recursivo liga los cálculos de las diferentes etapas y la solución final del problema se obtiene cuando se alcanza la última etapa
- En cada etapa se toma una decisión (Se realiza una acción)
- Una secuencia de decisiones es una ***Política***
- Cada etapa tiene uno o varios estados asociados a ella
- Un estado representa una condición en la que puede estar el sistema asociado con el problema se puede encontrar en alguna etapa
- El efecto de una decisión en cierta etapa es cambiar de un estado relacionado con la etapa actual a algún estado relacionado con la siguiente etapa.

Introducción

- Usualmente, se obtiene un resultado en la forma de un costo o de un beneficio cada vez que se toma una decisión.
- El objetivo es encontrar una política óptima, es decir aquella que obtiene el mayor beneficio.

Principio de optimalidad de Bellman (1957)

- Cualquiera que sea el estado inicial y la decisión inicial, las decisiones restantes deben conformar una política óptima respecto al estado resultante de la primera decisión
- Esto implica que la política óptima para las etápas restantes es independiente de la política utilizada en las etapas anteriores
- El conocimiento del estado actual del sistema encierra toda la información acerca del comportamiento pasado que se requiere para determinar una política óptima de ahí en adelante.
- Es la propiedad de Markov

El principo de Bellman en acción

- En base a este principio, se debe comenzar por encontrar las política óptima para cada estado de la última etapa y después moverse hacia atrás etapa por etapa de modo que en cada etapa se encuentre la mejor política de abandonar cada uno de los estados de la etapa en turno.

Ejemplo

Se cuenta con un recurso de X unidades a ser repartidas en N actividades,

Por ejemplo, $\$X$ Se debe distribuir en N proyectos

Una tabla $r_i(x)$ indica el beneficio obtenido al dedicar x unidades al proyecto i

Queremos maximizar $\sum_{i=1}^{N} r_i(x_i)$ sujeto a $\sum_{i=1}^{N} x_i = X$

$$0 \le x_i \le X$$

$$i = 1,2,\ldots, N$$

Solución

Para usar el principio de optimalidad, podemos ver la asignación del recurso a cada proyecto como una etapa.

Entonces, hay N etapas

Etapa 1: Asignar x_1 unidades al proyecto 1

Etapa 2: Asignar $x - x_1$ unidades al proyecto 2

Etc.

$v_k(x)$.- Función de valor óptimo, es el beneficio máximo que se puede obtener a partir de la actividad ***k*** hacia la actividad ***N*** dado que restan ***x*** unidades por asignarse

Solución (continuación)

Del principio de optimalidad obtenemos la recurrencia:

$$v_k(x) = \max\{r_k(x_k) + v_{k+1}(x - x_{k+1})\} \qquad x_k = 0,1,\ldots, x$$

La condición de frontera es: $v_N(x) = r_N(x)$

Y la solución es: $v_1(x)$

Ejemplo numérico

Sea ***X***=6 y ***N***=3

x_1	$r_1(x_1)$	x_2	$r_2(x_2)$	x_3	$r_3(x_3)$
0	0	0	0	0	0
1	3	1	2	1	1
2	6	2	4	2	3
3	9	3	6	3	5
4	12	4	9	4	8
5	16	5	11	5	12
6	16	6	13	6	13

Las condiciones de frontera son: $v_3(0) = 0,\ v_3(1) = 1,\ v_3(2) = 3,\ v_3(3) = 5,$
$v_3(4) = 8, v_3(5) = 12,\ v_3(6) = 13$

Que son esencialmente los valores de $r_3(x_3)$.

Sea $m_k(x)$ el valor de x_k que maximiza $r_k(x_k) + v_{k+1}(x - x_k)$ $x_k = 0, 1, \ldots, x$

Entonces, usando la recurrencia obtenemos:

$$v_2(0) = 0 \quad m_2(0) = 0$$

$$v_2(1) = \max[r_2(x_k) + v_3(1 - x_k)] = \max[r_2(0) + v_3(1), r_2(1) + v_3(0)]$$

$$= \max[0 + 1, 2 + 0] = 2 \quad m_2(1) = 1$$

Ejemplo numérico (continuación)

$$
\begin{aligned}
v_2(2) &= \max[r_2(0) + v_3(2), r_2(1) + v_3(1), r_2(2) + v_3(0)] \\
&= \max[0 + 3, 2 + 1, 4 + 0] = 4 \quad m_2(2) = 2 \\
v_2(3) &= \max[r_2(0) + v_3(3), r_2(1) + v_3(2), r_2(2) + v_3(1), r_2(3) + v_3(0)] \\
&= \max[0 + 5, 2 + 3, 4 + 1, 6 + 0] = 6 \quad m_2(3) = 3 \\
v_2(4) &= \max[r_2(0) + v_3(4), r_2(1) + v_3(3), r_2(2) + v_3(2), r_2(3) \\
&\quad + v_3(1), r_2(4) + v_3(0)] \\
&= \max[0 + 8, 2 + 5, 4 + 3, 6 + 1, 9 + 0] = 9 \quad m_2(4) = 4 \\
v_2(5) &= \max[r_2(0) + v_3(5), r_2(1) + v_3(4), r_2(2) + v_3(3), r_2(3) + v_3(2), r_2(4) \\
&\quad + v_3(1), r_2(5) + v_3(0)] \\
&= \max[0 + 12, 2 + 8, 4 + 5, 6 + 3, 9 + 1, 11 + 0] = 12 \quad m_2(5) = 0 \\
v_2(6) &= \max[r_2(0) + v_3(6), r_2(1) + v_3(5), r_2(2) + v_3(4), r_2(3) + v_3(3), r_2(4) \\
&\quad + v_3(2), r_2(5) + v_3(1), r_2(6) + v_3(0)] \\
&= \max[0 + 13, 2 + 12, 4 + 8, 6 + 5, 9 + 3, 11 + 1, 13 + 0] \\
&= 14 \quad m_2(6) = 1
\end{aligned}
$$

Ejemplo numérico (continuación)

Como iniciamos con 6 unidades, solo necesitamos calcular $v_1(6)$,

$$
\begin{aligned}
v_1(6) &= \max[r_1(0) + v_2(6), r_1(1) + v_2(5), r_1(2) + v_2(4), r_1(3) + v_2(3), r_1(4) \\
&\quad + v_2(2), r_1(5) + v_2(1), r_1(6) + v_2(0)] \\
&= \max[0 + 14, 2 + 12, 6 + 9, 9 + 6, 12 + 4, 16 + 2, 16 + 0] \\
&= 18 \quad m_1(6) = 5
\end{aligned}
$$

Entonces, el beneficio máximo que se puede obtener es de 18, las asignaciones que se deben hacer son:
Asignar 5 unidades al proyecto 1 ya que $m_1(6) = 5$,
Asignar 1 unidad al proyecto 2 ya que $m_2(1) = 1$.
No asignar ninguna unidad al proyecto 3

Se puede corroborar que $r_1(5) + r_2(1) = 18$

En Matlab:

```
 %V(i,j) tiene el valor que se obtendra asignando un presupuesto i a
 %los proyectos restantes (de j+1 en adelante)
 [N M]=size(R);
V=zeros(N,M);
%m(i,j) indica cuanto conviene asignarle a un proyecto dado el presupuesto i y

 %los proyectos restantes (de j+1 en adelante)
m=zeros(N,M-1);
% llena primero la ultima columna
for i=0:N-1
    V(i+1,M)=R(i+1,M);
end
for j=M-1:-1:1
    for i=0:N-1
        clear t;
        for k=0:i
            t(k+1)=R(k+1,j)+V(i-k+1,j+1);
        end
        [maximo,indice]=max(t);
        V(i+1,j)=maximo;
        m(i+1,j)=indice-1;
    end
end
y=V;
```

En Matlab (salida)

```
>> futureRewardDP

m =

    0    0
    1    1
    2    2
    3    3
    4    4
    5    0
    5    1


ans =

    0    0    0
    3    2    1
    6    4    3
    9    6    5
   12    9    8
   16   12   12
   18   14   13
```

Procesos de Markov con recompensa

- Un MRP es un proceso de Markov que se forma mediante una cadena de Markov y una Matriz de recompensas

$$R = \begin{bmatrix} r_{11} & r_{11} & \cdots & r_{1N} \\ r_{21} & r_{22} & \cdots & r_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ r_{N1} & r_{N2} & \cdots & r_{NN} \end{bmatrix}$$

r_{ij} es la recompensa inmediata que recibe el proceso cuando estando en el estado i, haga una transición al estado j, la recompensa puede ser positiva o negativa

Procesos de Markov con recompensa

Sea $v_n(i)$ el valor esperado de ingresos que se obtendrán en las próximas n transiciones dado que actualmente se encuentra en el estado i.

$$
\begin{aligned}
v_n(i) &= E[\Re_n + \Re_{n-1} + \cdots + \Re_1 + \Re_0 | s_n = i] \\
&= E[\{\Re_n | s_n = i\} + \{\Re_{n-1} + \cdots + \Re_1 + \Re_0\} | s_n = i] \\
&= E[\Re_n | s_n = i] + E[\Re_{n-1} + \cdots + \Re_1 + \Re_0 | s_n = i] \\
&= \sum_{j=1}^{N} p_{ij} r_{ij} + \sum_{j=1}^{N} p_{ij} v_{n-1}(j)
\end{aligned}
$$

$\Re_n$ es la recompensa inmediata obtenida cuando falten n transiciones por efectuarse

s_n es el estado actual

Definiendo:

$$q_i = \sum_{j=1}^{N} p_{ij} r_{ij}$$

Sustituyendo:

$$v_n(i) = q_i + \sum_{j=1}^{N} p_{ij} v_{n-1}(j)$$

Procesos de decisión de Markov

- Son una extensión de los MRP y a la vez de la Programación dinámica
- Un tomador de decisiones o ***agente*** puede influenciar el estado del sistema realizando una secuencia de ***acciones*** que logran que el sistema optimice su desempeño de acuerdo a algún criterio.
- El agente observa el estado del sistema en momentos específicos y reúne la información necesaria para tomar acciones.
- Cada acción realizada por el agente tiene un costo o una recompensa
- Las acciones afectan el estado del sistema e influyen entonces en decisiones futuras
- En conclusión, al realizar una acción, el agente incurre en un costo inmediato y el sistema cambia de estado de acuerdo a una distribución de probabilidades de transición entre estados
- El costo inmediato depende del estado y de la acción seleccionada

Procesos de decisión de Markov

- T.- Conjunto de momentos de decisión
- Por un lado deseamos bajo costo inmediato y por otro lado deseamos bajo costo de largo plazo (***Horizonte***)
- Una ***política*** establece que acción se debe tomar en cada momento de decisión.

Una política es $D = (d_1, d_2, \ldots, d_{N-1})$ donde d_t es la acción que se debe tomar en el momento de decisión $t \in T$

Las políticas pueden ser estacionarias o no-estacionarias

Ej de política estacionaria: Apostar $2 cada vez que se obtenga cara y $1 cada vez que salga cruz (los estados corresponden con el resultado obtenido al arrojar una moneda)

En una política no estacionaria, se pueden tomar diferentes acciones al estar en cierto estado En diferentes momentos de decisión (dado que el horizonte es diferente)

Procesos de decisión de Markov

Un MDP es un tuple (S, A, R, P)

donde S es el conjunto de N estados

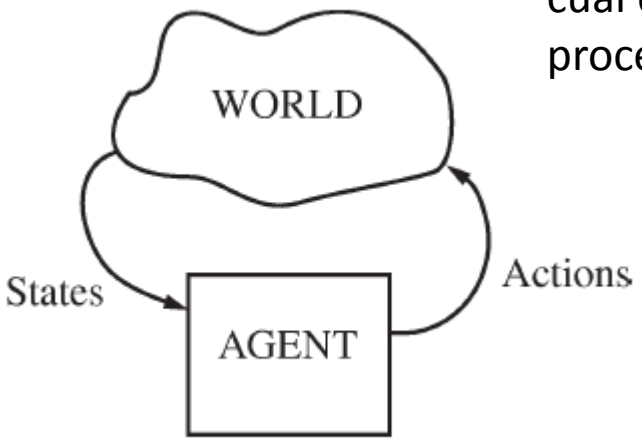
A es el conjunto de K acciones

R es la matriz de recompensas la cual depende de la acción tomada. La recompensa asociada con la transición del estado i al estado j cuando se toma la acción a

$$r_{i,j}(a)$$

P Es la matriz de probabilidad de transición entre estados la cual depende de la acción . La probabilidad de que el proceso cambie del estado i al j cuando se toma la acción a

$$p_{i,j}(a)$$



Procesos de decisión de Markov

Para estos sistemas se cumple:

$$P[S_{n+1} = j | S_0, a_0, S_1, a_1, \ldots, S_n = i, a_n = a]$$
$$= P[S_{n+1} = j | S_n = i, a_n = a] = p_{ij}(a)$$

Las funciones de probabilidad de transición y de recompensa dependen solo del último estado y de la acción

$$\sum_j p_{ij}(a) = 1$$ Para cada estado i y acción $a \in A$

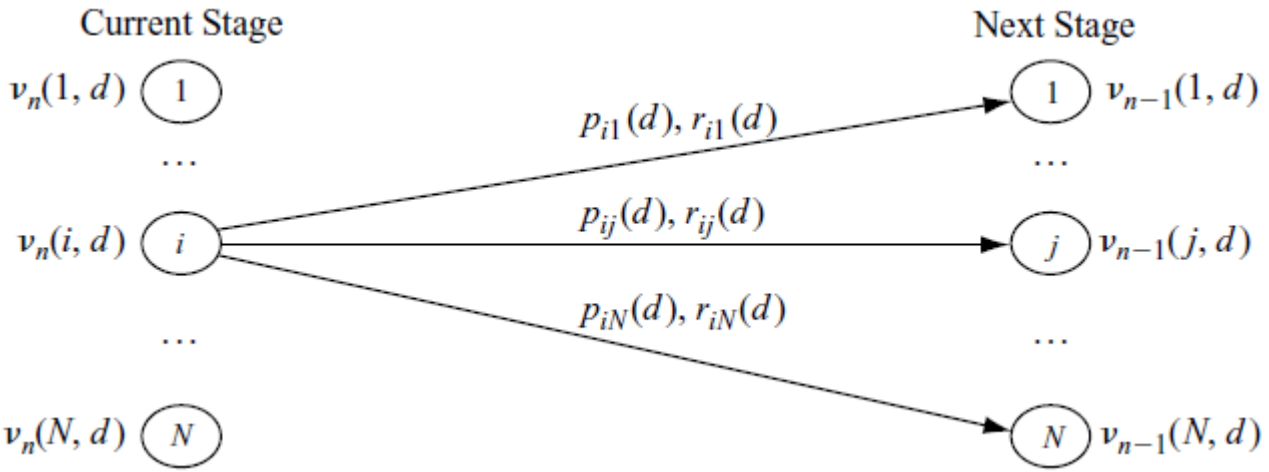
Política

- Una política especifica la acción que se debe tomar en cada estado
- Una Política ***D*** es un mapeo de ***S*** a ***A***
- El objetivo es maximizar el valor esperado de la suma de recompensas a largo plazo
- Definimos una ***política óptima*** como aquella que maximiza el valor esperado de recompensa total partiendo de un estado ***i*** para cierto número de transiciones ***n***

La función valor

Sea $v_n(i, d)$ la recompensa total esperada para los próximos ***n*** estados dado que el proceso está en el estado ***i*** y se utiliza la política ***d***

a $v_n(i, d)$ se le conoce como ***la función valor***



$$v_n(i, d) = \sum_{j=1}^{N} p_{ij}(d) r_{ij}(d) + \sum_{j=1}^{N} p_{ij}(d) v_{n-1}(j, d)$$

$$= q_i(d) + \sum_{j=1}^{N} p_{ij}(d) v_{n-1}(j, d)$$

donde: $q_i(d) = \sum_{j=1}^{N} p_{ij}(d) r_{ij}(d)$

El problema de determinar la política óptima

Aplicando el principio de optimalidad de Bellman, la recompensa óptima obtenida despues de n transiciones partiendo del estado i es:

$$v_n(i) = \max_d \left\{ q_i(d) + \sum_{j=1}^{N} p_{ij}(d) v_{n-1}(j) \right\} \qquad i = 1, 2, \ldots, N$$

MDP con descuento

En economía como en muchas áreas de la ingeniería, un monto obtenido en el momento vale mas que el mismo monto a obtenerse en el futuro, para modelar esto se utiliza un ***factor de descuento*** $0 \leq \beta < 1$

$$
\begin{aligned}
v_n(i,d,\beta) &= E[\Re_n + \beta\Re_{n-1} + \beta^2\Re_{n-2} + \cdots + \beta^{n-1}\Re_1 + \beta^n\Re_0 | s_n = i] \\
&= E[\{\Re_n | s_n = i\} + \beta\{\Re_{n-1} + \beta\Re_{n-2} + \cdots + \beta^{n-2}\Re_1 \\
&\quad + \beta^{n-1}\Re_0\} | s_n = i] \\
&= E[\Re_n | s_n = i] + \beta E[\Re_{n-1} + \beta\Re_{n-2} + \cdots + \beta^{n-2}\Re_1 \\
&\quad + \beta^{n-1}\Re_0 | s_n = i] \\
&= \sum_{j=1}^{N} p_{ij}(d) r_{ij}(d) + \beta \sum_{j=1}^{N} p_{ij}(d) v_{n-1}(j,d,\beta) \\
&= q_i(d) + \beta \sum_{j=1}^{N} p_{ij}(d) v_{n-1}(j,d,\beta)
\end{aligned}
$$

Métodos de solución

- Solucionar un MDP equivale a encontrar una secuencia de acciones que maximizan la función valor
- El método ***valor-iteración*** se utiliza para resolver el problema para un horizonte finito
- El método ***política-iteración*** se utiliza para problemas de horizonte infinito
- El método de ***programación lineal*** también sirve para problemas de horizonte infinito
- Al método de valor-iteración también se le conoce como de ***aproximaciones sucesivas***
- A diferencia de los métodos de política-iteración y de programación lineal, el método de valor-iteración no requiere solucionar un sistema de ecuaciones simultáneas en cada iteración por lo que se considera el método mas veloz y simple.

El método valor-iteración

Este método calcula recursivamente una secuencia de funciones valor aproximando sucesivamente el valor óptimo. Es una extensión de la técnica utilizada para resolver el problema determinista de programación dinámica y usa la relación recursiva:

$$v_n(i,d,\beta) = \sum_{j=1}^{N} p_{ij}(d) r_{ij}(d) + \sum_{j=1}^{N} p_{ij}(d) \beta v_{n-1}(j,d,\beta)$$

$$= q_i(d) + \beta \sum_{j=1}^{N} p_{ij}(d) v_{n-1}(j,d,\beta)$$

$$v_n(i,\beta) = \max_d \left\{ q_i(d) \sum_{j=1}^{N} p_{ij}(d) v_{n-1}(j,\beta) \right\} \quad i = 1, 2, \ldots, N$$

El método valor-iteración

Inicialización:

$$v_0(1, d, \beta) = v_0(2, d, \beta) = \ldots = v_0(N, d, \beta) = 0$$

$$v_1(i, \beta) = \max_d \{q_i(d)\} \quad i = 1, 2, \ldots, N$$

Ahora que ya se cuenta con la recompensa esperada en la etapa 1 (cuando falta solo una transición), para cada estado ***i***, podemos obtener la recompensa total esperada en la etapa 2 (cuando falten 2 transiciones):

$$v_2(i, \beta) = \max_d \left\{ q_i(d) + \sum_{j=1}^{N} p_{ij}(d) v_1(j, \beta) \right\}$$

El proceso continúa y la solución al problema la tendremos en $v_T(i)$

$$v_T \leftarrow v_{T-1} \leftarrow \ldots \leftarrow v_2 \leftarrow v_1 \leftarrow v_0$$

Ejemplo

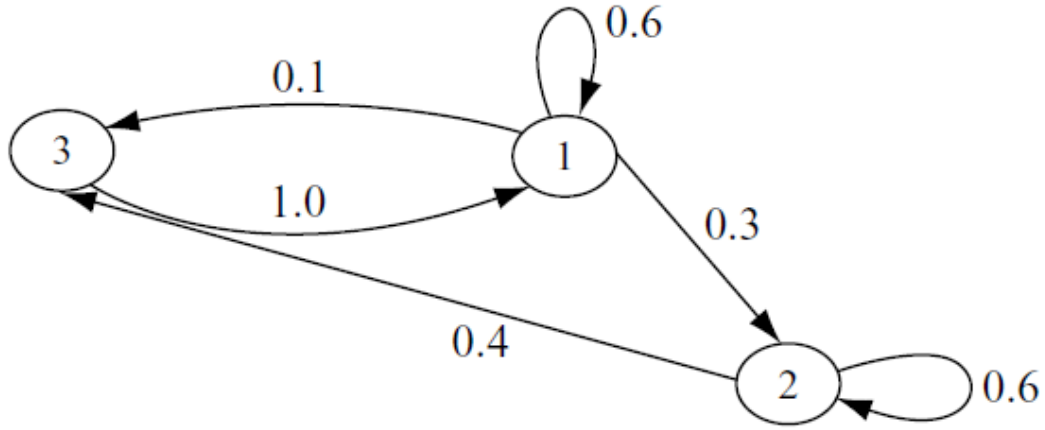
- Una compañía realiza inspección diaria de su equipo, al final del día el cual puede estar en estado Bueno, Aceptable y Malo.
- Si su estado es bueno, al día siguiente su estado continuará siendo bueno con probabilidad de 0.6, aceptable con probabilidad de 0.3 y malo de 0.1
- Si el estado es Aceptable, al día siguiente el estado será aceptable o malo con probabilidades de 0.6 y 0.4 respectivamente
- Si el estado es malo al día siguiente continuará en el mismo estado
- Las posibles acciones son:
  1. No hacer nada
  2. Dar mantenimiento, lo cual hará que el equipo al día siguiente en buen estado o en estado aceptable con probabilidad igual, el costo del mantenimiento es de $500
  3. Reemplazar el equipo con lo cual el equipo estará en buen estado, el costo del equipo es de $2,000
- Cuando el equipo está en buen estado la compañía gana $1000 diarios, si el equipo está en estado aceptable gana $500 diarios y si está en mal estado pierde $500 diarios

Ejemplo

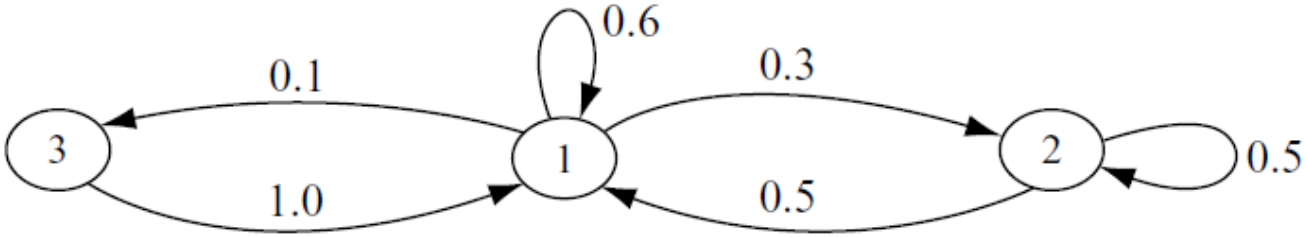
- Solo se considerarán las siguientes políticas
  1. Reemplazar el equipo solo si está en mal estado
  2. Reemplazar el equipo cuando está en mal estado y darle mantenimiento cuando está en estado aceptable
  3. Reemplazar el equipo si está en mal estado o en estado aceptable
- El problema es determinar el costo óptimo de operación del equipo para T=4

Solución

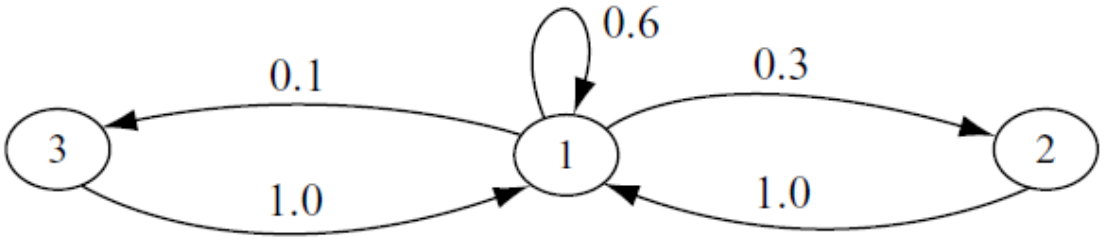
– Sean los estados: 1=Bueno; 2=Aceptable: 3=Malo



Política 1



Política 2



Política 3

Solución

- Asumamos que los costos son ganancias negativas y expresemos las ganancias en unidades de $500
- Para la política ***d*** la distribución de transición entre estados es ***P(d),*** la matriz de recompensas es ***R(d)*** y el valor esperado de recompensa inmediata es ***Q(d)***

$$P(1) = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0 & 0.6 & 0.4 \\ 1 & 0 & 0 \end{bmatrix} \quad R(1) = \begin{bmatrix} 2 & 1 & -1 \\ 0 & 1 & -1 \\ -2 & 0 & 0 \end{bmatrix} \quad Q(1) = \begin{bmatrix} 1.4 \\ 0.2 \\ -2 \end{bmatrix}$$

$$P(2) = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0.5 & 0.5 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad R(2) = \begin{bmatrix} 2 & 1 & -1 \\ 1 & 0 & 0 \\ -2 & 0 & 0 \end{bmatrix} \quad Q(2) = \begin{bmatrix} 1.4 \\ 0.5 \\ -2 \end{bmatrix}$$

$$P(3) = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad R(3) = \begin{bmatrix} 2 & 1 & -1 \\ -2 & 0 & 0 \\ -2 & 0 & 0 \end{bmatrix} \quad Q(3) = \begin{bmatrix} 1.4 \\ -2 \\ -2 \end{bmatrix}$$

Etapa 1

Comenzamos con $v_0(i,d) = 0$

Por lo tanto $v_1(i,d) = q_i(d)$

	$v_1(i,d) = q_i(d)$			Optimal Solution	
i	$d=1$	$d=2$	$d=3$	$v_1(i)$	d^*
1	1.4	1.4	1.4	1.4	1, 2, 3
2	0.2	0.5	−2	0.5	2
3	−2	−2	−2	−2	1, 2, 3

Etapa 2

	$v_2(i,d) = q_i(d) + p_{i1}(d)v_1(1) + p_{i2}(d)v_1(2) + p_{i3}(d)v_1(3)$			Optimal Solution	
i	$d=1$	$d=2$	$d=3$	$v_2(i)$	d^*
1	2.19	2.19	2.19	2.19	1, 2, 3
2	-0.3	1.45	-0.6	1.45	2
3	−0.6	−0.6	−0.6	−0.6	1, 2, 3

$$v_2(1,1) = q_1(1) + p_{1,1}(1)v_1(1) + p_{1,2}(1)v_1(2) + p_{1,3}(1)v_1(3)$$
$$v_2(1,1) = 1.4 + (0.6)(1.4) + (0.3)(0.5) + (0.1)(-2) = 2.19$$

$$v_2(2,1) = q_2(1) + p_{2,1}(1)v_1(1) + p_{2,2}(1)v_1(2) + p_{2,3}(1)v_1(3)$$
$$v_2(2,1) = 0.2 + (0)(1.4) + (0.6)(0.5) + (0.4)(-2) = -0.3$$

$$v_2(3,1) = q_3(1) + p_{3,1}(1)v_1(1) + p_{3,2}(1)v_1(2) + p_{3,3}(1)v_1(3)$$
$$v_2(3,1) = -2 + (1)(1.4) + (0)(0.5) + (0)(-2) = -0.6$$

Etapa 3

	$v_3(i,d) = q_i(d) + p_{i1}(d)v_2(1) + p_{i2}(d)v_2(2) + p_{i3}(d)v_2(3)$			Optimal Solution	
i	$d=1$	$d=2$	$d=3$	$v_3(i)$	d^*
1	3.089	3.089	3.089	3.089	1, 2, 3
2	0.83	2.32	0.19	2.32	2
3	0.19	0.19	0.19	0.19	1, 2, 3

Etapa 4

	$v_4(i,d) = q_i(d) + p_{i1}(d)v_3(1) + p_{i2}(d)v_3(2) + p_{i3}(d)v_3(3)$			Optimal Solution	
i	$d=1$	$d=2$	$d=3$	$v_4(i)$	d^*
1	3.9684	3.9684	3.9684	3.9684	1, 2, 3
2	1.668	3.2045	1.089	3.2045	2
3	1.089	1.089	1.089	1.089	1, 2, 3

En Matlab

```
P=      [0.6 0.3 0.1;
         0  0.6 0.4;
         1   0   0];
P(:,:,2)=[0.6 0.3 0.1;
        0.5 0.5  0;
         1    0   0];
P(:,:,3)=[0.6 0.3 0.1;
         1    0   0;
         1    0   0];
R=       [ 2 1 -1;
         0 1 -1;
         -2 0  0];
R(:,:,2)=[ 2 1 -1;
         1 0  0 ;
         -2 0  0];
R(:,:,3)=[ 2 1 -1;
         -2 0 0;
%           1 0 0;
         -2 0  0];
```

```
for i=1:3
    for d=1:3
      q(i,d)=sum(P(i,:,d).*R(i,:,d));
   end
end
V=zeros(3,3);
for i=1:3  %3 estados
    for d=1:3  %3 politicas
      V(i,d)=q(i,d);
    end
end
for i=1:3
    v(i)=max(V(i,:));
end
v
```

```
for t=2:4 % 4 etapas
   for d=1:3
      for i=1:3
         suma=0;
         for j=1:3
            suma=suma+P(i,j,d)*v(j);
         end
         V(i,d)=q(i,d)+suma;
      end
   end
   V
    for i=1:3
       v(i)=max(V(i,:));
    end
    v
end
```

POMDP

Es un tuple $(S, A, \Omega, P, \Phi, R)$

donde

S,A, P y ***R*** describen un proceso de decisión de Markov

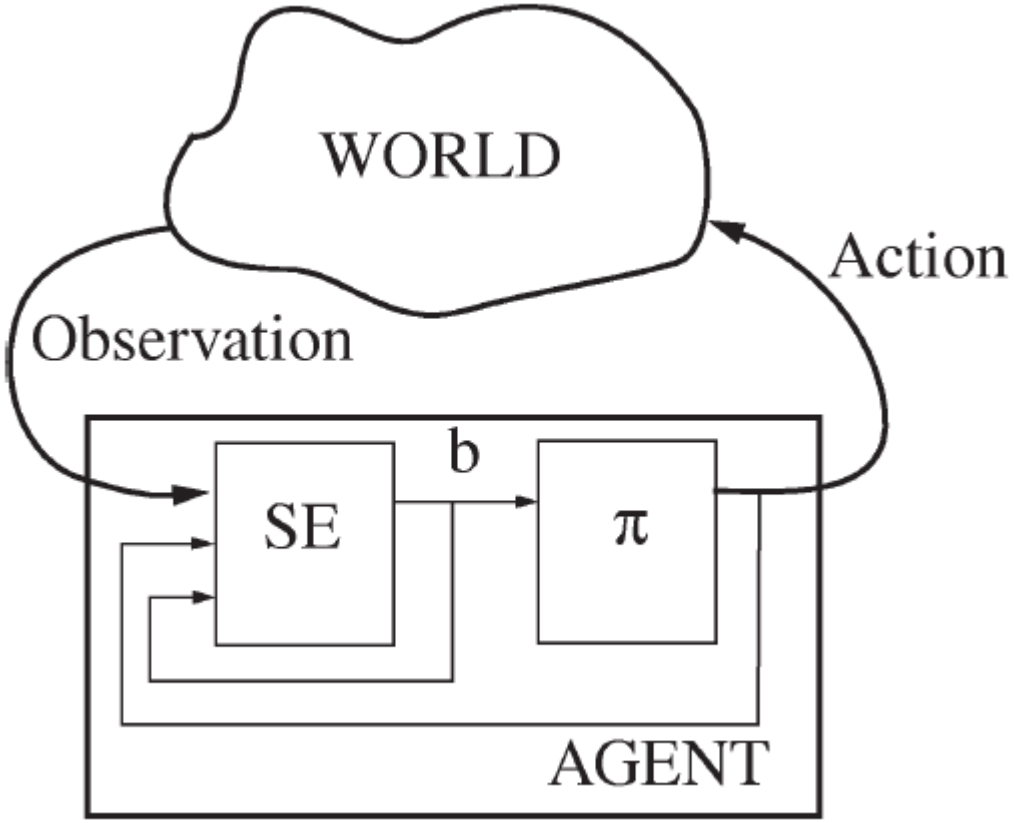
Ω es un conjunto finito de observaciones que el agente puede experimentar acerca del mundo

Φ es la función de observaciones que por cada acción y estado resultante entrega una distribución de probabilidades de observaciones $\phi_{ij}(a) = P[\Omega_t = o_j | S_t = s_i, A_{t-1} = a]$

POMDP

- Es un MDP en el cual el agente no puede observar el estado actual, pero en lugar de eso puede ver una observación basada en la acción y en el estado resultante
- El objetivo del agente sigue siendo maximizar la recompensa futura esperada con descuento

POMDP



El agente de un POMDP se puede descomponer en un estimador de estados y una política

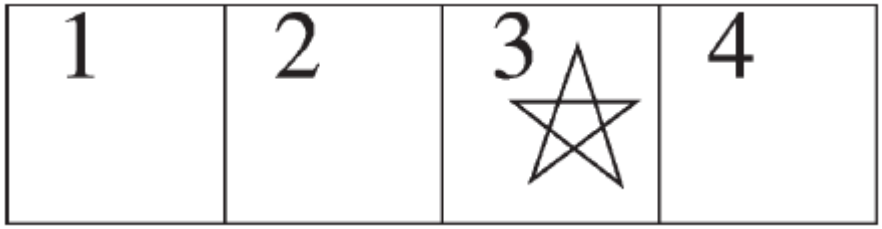
El control de un POMDP

- El agente tiene un estimador de estado que es responsable de mantener el *belief state* ***b*** basándose en la última acción, la observación actual y el ***b*** previo
- La política es responsable de generar acciones solo que es función de ***b*** en lugar del estado del mundo

Qué es el *belief state*?

- Una opción hubiera sido el estado con mayor probabilidad. Pero entonces el agente no estaría tomando en cuenta la incertidumbre
- Le llamamos “belief state” a la distribución de probabilidades de estados.
- Provee una base para actuar bajo incertidumbre
- Asegura una estadística suficiente acerca del pasado hasta el “belief state” inicial
- El proceso se puede considerar markoviano respecto a los belief sates

Ejemplo



Cuatro estados, dos posibles observaciones, una en el estado 3 y la otra en los otros estados. Dos acciones (Izquierda o Derecha). Las acciones son exitosas con probabilidad 0.9 y cuando fallan el movimiento se da en la dirección opuesta. Si el movimiento no es posible, el agente no se mueve. Inicialmente el agente se encuentra con la misma probabilidad en los estados 1,2 o 4

$$b=[0.333 \;\; 0.333 \;\; 0.000 \; 0.333]$$

Si el agente ejecuta la acción “Derecha” y no observa la estrella, entonces :

$$b=[0.100 \; 0.450 \;\; 0.000 \; 0.450]$$

Otra vez (acción “Derecha” y no observa estrella)

$$b=[0.100 \; 0.164 \; 0.000 \; 0.736]$$

Calculando el *belief state*

$$b_t(s_k) = P[S_t = s_k | \Omega_t = o_m, A_{t-1} = a, \Omega_{t-1}, \ldots, A_0] = P[s_k | o_m, a, b_{t-1}(s_j)]$$

$$= \frac{P[s_k, o_m, a, b_{t-1}(s_j)]}{P[o_m, a, b_{t-1}(s_j)]} = \frac{P[o_m | s_k, a, b_{t-1}(s_j)] P[s_k, a, b_{t-1}(s_j)]}{P[o_m | a, b_{t-1}(s_j)] P[a, b_{t-1}(s_j)]}$$

$$= \frac{P[o_m | s_k, a, b_{t-1}(s_j)] P[s_k | a, b_{t-1}(s_j)] P[a, b_{t-1}(s_j)]}{P[o_m | a, b_{t-1}(s_j)] P[a, b_{t-1}(s_j)]}$$

$$= \frac{P[o_m | s_k, a, b_{t-1}(s_j)] P[s_k | a, b_{t-1}(s_j)]}{P[o_m | a, b_{t-1}(s_j)]}$$

$$= \frac{P[o_m | s_k, a] \sum_{s \in S} P[s_k | a, b_{t-1}(s), s] P[s | a, b_{t-1}(s)]}{P[o_m | a, b_{t-1}(s_j)]}$$

$$= \frac{P[o_m | s_k, a] \sum_{s_j \in S} P[s_k | a, s_j] b_{t-1}(s_j)}{P[o_m | a, b_{t-1}(s_j)]} = \frac{\phi_{km}(a) \sum_{s_j \in S} p_{jk}(a) b_{t-1}(s_j)}{P[o_m | a, b_{t-1}(s_j)]}$$

El denominador puede considerarse como un factor de normalización

La función de actualizacion del belief state

- Dado un belief state podemos determinar el belief state sucesor para una determinada acción y una determinada observación

$$b_t(s) = \tau(b_{t-1}, A_{t-1}, \Omega_t)$$

Solución de un POMDP

Un POMDP puede considerarse un MDP definido sobre el espacio del *belief state.* En particular, la recompensa asociada con la acción *a* y *belief state b* es:

$$r(b,a) = \sum_{s_i \in S} \sum_{s_j \in S} r_{ij}(a)\, p_{ij}(a) b(s_i) = \sum_{s_i \in S} r_i(a) b(s_i)$$

$$v_0^*(b) = \max_{a \in A} r(b,a)$$

$$v_t^*(b) = \max_{a \in A} \left\{ r(b,a) + \beta \sum_{o \in \Omega} P[o|b,a] v_{t-1}^*(\tau(b,a,o)) \right\}$$

$$= \max_{a \in A} \left\{ r(b,a) + \beta \sum_{s_k \in S} P[s_k|s_i,a] \sum_{o_j \in \Omega} P[o_j|s_k,a] b(s_i) v_{t-1}^*(\tau(b,a,o)) \right\}$$

$$= \max_{a \in A} \left\{ r(b,a) + \beta \sum_{s_k \in S} p_{ik}(a) \sum_{o_j \in \Omega} \phi_{kj}(a) b(s_i) v_{t-1}^*(\tau(b,a,o)) \right\}$$

Determinando la política óptima

- Una política es una secuencia de decisiones
- La política óptima es la que maximiza el valor esperado de recompensa a futuro con descuento
- La política óptima estará dada por:

$$d^*(b_t) = \arg \max_{a \in A} \left[r(b,a) + \beta \sum_{o \in \Omega} P[o|b,a] v^*_{t-1}(\tau(b,a,o)) \right]$$

$$= \arg \max_{a \in A} \left[r(b,a) + \beta \sum_{s_k \in S} p_{ik}(a) \sum_{o_j \in \Omega} \phi_{kj}(a) b(s_i) v^*_{t-1}(\tau(b,a,o)) \right]$$

Ejemplo. Robot de vigilancia

- Estados

$$S = \{s_1 = (0,0, X_1), s_2 = (0,1, X_1), s_3 = (1,0, X_1), s_4 = (1,1, X_1),$$
$$s_5 = (0,0, X_2), s_6 = (0,1, X_2), s_7 = (1,0, X_2), s_8 = (1,1, X_2)\}$$

Formato de estados $(f_1, f_2 \ldots f_n, X_l)$

f_i Variable booleana que indica fuego en el área X_i

X_l Área en la que se encuentra el robot

- Estado Inicial. $s_1 = (0,0, X_1)$ $b(s_1) = 1$ $b(s) = 0$ for all other $s \in S$.
- Acciones $a_1 = GO(X_1)$ $a_2 = GO(X_2)$
- Observaciones

$$\Omega = \{o_1 = (f_1 = 0), o_2 = (f_1 = 1), o_3 = (f_2 = 0), o_4 = (f_2 = 1)\}$$

Transiciones $T : S \times A \rightarrow \Pi(S)$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
s_1, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_2, a_1	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0
s_3, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_4, a_1	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0
s_5, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_6, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_7, a_1	0.0	0.0	0.2	0.8	0.0	0.0	0.0	0.0
s_8, a_1	0.0	0.0	0.2	0.8	0.0	0.0	0.0	0.0
s_1, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_2, a_2	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5
s_3, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_4, a_2	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5
s_5, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_6, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_7, a_2	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.8
s_8, a_2	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.8

Recompensas

	a_1	a_2
s_1	0.5	0.8
s_2	0.5	1.0
s_3	0.5	0.8
s_4	0.5	1.0
s_5	0.5	0.8
s_6	0.5	0.8
s_7	1.0	0.8
s_8	1.0	0.8

Función de observaciones

$$O = S \times A \rightarrow \Pi(\Omega)$$

	o_1	o_2	o_3	o_4
$s_1, a_1 \vee a_2$	1.0	0.0	0.0	0.0
$s_2, a_1 \vee a_2$	1.0	0.0	0.0	0.0
$s_3, a_1 \vee a_2$	0.0	1.0	0.0	0.0
$s_4, a_1 \vee a_2$	0.0	1.0	0.0	0.0
$s_5, a_1 \vee a_2$	0.0	0.0	1.0	0.0
$s_6, a_1 \vee a_2$	0.0	0.0	0.0	1.0
$s_7, a_1 \vee a_2$	0.0	0.0	1.0	0.0
$s_8, a_1 \vee a_2$	0.0	0.0	0.0	1.0

$$\Omega = \{o_1 = (f_1 = 0), o_2 = (f_1 = 1), o_3 = (f_2 = 0), o_4 = (f_2 = 1)\}$$

$$S = \{s_1 = (0,0,X_1), s_2 = (0,1,X_1), s_3 = (1,0,X_1), s_4 = (1,1,X_1),$$
$$s_5 = (0,0,X_2), s_6 = (0,1,X_2), s_7 = (1,0,X_2), s_8 = (1,1,X_2)\}$$

En este ejemplo en particular no importa la acción, la probabilidad de ver una observación depende solo del estado

Horizonte de 1

El robot comienza con $b(s_1) = 1$

$$R(s_1, a_1) = 0.5 \qquad R(s_1, a_2) = 0.8$$

Por lo cual la acción elegida primero es a_2

Suponga que la observación recibida es o_3

Calculemos ahora el nuevo belief *b'*

$$P(s' | b, a_2) = \sum_{s \in S} T(s, a_2, s') b(s)$$

$$P(s_1 | b, a_2) = \sum_{s \in S} T(s, a_2, s_1) b(s)$$

$$P(s_1 | b, a_2) = T(s_1, a_2, s_1) b(s_1) + T(s_2, a_2, s_1) b(s_2) + \ldots + T(s_8, a_2, s_1) b(s_8)$$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s' \mid b, a_2)$	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4

Calculamos el nuevo *belief*

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s'\|b, a_2)$	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4

	o_1	o_2	o_3	o_4
$s_1, a_1 \vee a_2$	1.0	0.0	0.0	0.0
$s_2, a_1 \vee a_2$	1.0	0.0	0.0	0.0
$s_3, a_1 \vee a_2$	0.0	1.0	0.0	0.0
$s_4, a_1 \vee a_2$	0.0	1.0	0.0	0.0
$s_5, a_1 \vee a_2$	0.0	0.0	1.0	0.0
$s_6, a_1 \vee a_2$	0.0	0.0	0.0	1.0
$s_7, a_1 \vee a_2$	0.0	0.0	1.0	0.0
$s_8, a_1 \vee a_2$	0.0	0.0	0.0	1.0

Con esta tabla y la función de observaciones podemos determinar

$$P(o|b, a_2) = \Sigma_{s' \in S} O(s', a_2, o) P(s'|b, a_2)$$

	o_1	o_2	o_3	o_4
$P(o\|b, a_2)$	0.0	0.0	0.2	0.8

De donde podemos finalmente obtener el nuevo belief mediante

$$b'(s') = \frac{O(s', a_2, o_3) P(s'|b, a_2)}{P(o_3|b, a_2)}$$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b'(s')$	0.0	0.0	0.0	0.0	0.5	0.0	0.5	0.0

Decidir la siguiente acción a tomar

Determinamos la recompensa esperada para las posibles acciones

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b'(s')$	0.0	0.0	0.0	0.0	0.5	0.0	0.5	0.0

	a_1	a_2
s_1	0.5	0.8
s_2	0.5	1.0
s_3	0.5	0.8
s_4	0.5	1.0
s_5	0.5	0.8
s_6	0.5	0.8
s_7	1.0	0.8
s_8	1.0	0.8

$$ER(a_1) = b'(s_5)R(s_5, a_1) + b'(s_7)R(s_7, a_1) = 0.5 \cdot 0.5 + 0.5 \cdot 1.0 = 0.75$$

$$ER(a_2) = b'(s_5)R(s_5, a_2) + b'(s_7)R(s_7, a_2) = 0.5 \cdot 0.8 + 0.5 \cdot 0.8 = 0.8$$

De ahí que el robot se decidirá por la acción:

$$a_2 = GO(X_2)$$

Supongamos que se recibe ahora la observación o_4

Si después de tomar la acción a_2 se recibe la observación o_4

Repetimos el procedimiento

Hay que determinar el nuevo *belief b''*

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b'(s')$	0.0	0.0	0.0	0.0	0.5	0.0	0.5	0.0

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
s_1, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_2, a_1	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0
s_3, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_4, a_1	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0
s_5, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_6, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_7, a_1	0.0	0.0	0.2	0.8	0.0	0.0	0.0	0.0
s_8, a_1	0.0	0.0	0.2	0.8	0.0	0.0	0.0	0.0
s_1, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_2, a_2	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5
s_3, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_4, a_2	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5
s_5, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_6, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_7, a_2	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.8
s_8, a_2	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.8

$$P(s'|b',a_2) = \sum_{s \in S} T(s,a_2,s')b'(s)$$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s'\|b',a_2)$	0.0	0.0	0.0	0.0	0.05	0.2	0.15	0.6

Calculando el nuevo *belief (b'')*

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s'\|b', a_2)$	0.0	0.0	0.0	0.0	0.05	0.2	0.15	0.6

	o_1	o_2	o_3	o_4
$s_1, a_1 \vee a_2$	1.0	0.0	0.0	0.0
$s_2, a_1 \vee a_2$	1.0	0.0	0.0	0.0
$s_3, a_1 \vee a_2$	0.0	1.0	0.0	0.0
$s_4, a_1 \vee a_2$	0.0	1.0	0.0	0.0
$s_5, a_1 \vee a_2$	0.0	0.0	1.0	0.0
$s_6, a_1 \vee a_2$	0.0	0.0	0.0	1.0
$s_7, a_1 \vee a_2$	0.0	0.0	1.0	0.0
$s_8, a_1 \vee a_2$	0.0	0.0	0.0	1.0

$$P(o|b, a_2) = \Sigma_{s' \in S} O(s', a_2, o) P(s'|b, a_2)$$

	o_1	o_2	o_3	o_4
$P(o\|b', a_2)$	0.0	0.0	0.2	0.8

$$b''(s') = \frac{O(s', a_2, o_4) P(s'|b', a_2)}{P(o_4|b', a_2)}$$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b''(s')$	0.0	0.0	0.0	0.0	0.0	0.25	0.0	0.75

Decidir la siguiente acción a tomar

Determinamos la recompensa esperada para las posibles acciones

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b''(s')$	0.0	0.0	0.0	0.0	0.0	0.25	0.0	0.75

	a_1	a_2
s_1	0.5	0.8
s_2	0.5	1.0
s_3	0.5	0.8
s_4	0.5	1.0
s_5	0.5	0.8
s_6	0.5	0.8
s_7	1.0	0.8
s_8	1.0	0.8

$$ER(a_1) = b''(s_6)R(s_6, a_1) + b''(s_8)R(s_8, a_1) = 0.25 \cdot 0.5 + 0.75 \cdot 1.0 = 0.875$$

$$ER(a_2) = b''(s_6)R(s_6, a_2) + b''(s_8)R(s_8, a_2) = 0.25 \cdot 0.8 + 0.75 \cdot 0.8 = 0.8$$

De ahí que el robot se decidirá por la acción:

$$a_1 = GO(X_1)$$

Supongamos que se recibe ahora la observación o_1

Si después de tomar la acción a_1 se recibe la observación o_1

Repetimos el procedimiento

Hay que determinar el nuevo *belief* *b’’’*

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b''(s')$	0.0	0.0	0.0	0.0	0.0	0.25	0.0	0.75

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
s_1, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_2, a_1	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0
s_3, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_4, a_1	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0
s_5, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_6, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_7, a_1	0.0	0.0	0.2	0.8	0.0	0.0	0.0	0.0
s_8, a_1	0.0	0.0	0.2	0.8	0.0	0.0	0.0	0.0
s_1, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_2, a_2	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5
s_3, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_4, a_2	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5
s_5, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_6, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_7, a_2	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.8
s_8, a_2	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.8

$$P(s'|b'',a_1)=\sum_{s\in S}T(s,a_1,s')b''(s)$$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s'\|b'',a_1)$	0.025	0.1	0.175	0.7	0.0	0.0	0.0	0.0

Calculando el nuevo *belief (b’’’)*

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s'\|b'', a_1)$	0.025	0.1	0.175	0.7	0.0	0.0	0.0	0.0

	o_1	o_2	o_3	o_4
$s_1, a_1 \vee a_2$	1.0	0.0	0.0	0.0
$s_2, a_1 \vee a_2$	1.0	0.0	0.0	0.0
$s_3, a_1 \vee a_2$	0.0	1.0	0.0	0.0
$s_4, a_1 \vee a_2$	0.0	1.0	0.0	0.0
$s_5, a_1 \vee a_2$	0.0	0.0	1.0	0.0
$s_6, a_1 \vee a_2$	0.0	0.0	0.0	1.0
$s_7, a_1 \vee a_2$	0.0	0.0	1.0	0.0
$s_8, a_1 \vee a_2$	0.0	0.0	0.0	1.0

$$P(o\,|\,b'', a_1) = \sum_{s' \in S} O(s', a_1, o) P(s'|\,b'', a_1)$$

	o_1	o_2	o_3	o_4
$P(o\|b'', a_1)$	0.125	0.875	0.0	0.0

$$b'''(s') = \frac{O(s', a_1, o_1) P(s'|\,b'', a_1)}{P(o_1\,|\,b'', a_1)}$$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b'''(s')$	0.2	0.8	0.0	0.0	0.0	0.0	0.0	0.0

Decidir la siguiente acción a tomar

Determinamos la recompensa esperada para las posibles acciones

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b'''(s')$	0.2	0.8	0.0	0.0	0.0	0.0	0.0	0.0

	a_1	a_2
s_1	0.5	0.8
s_2	0.5	1.0
s_3	0.5	0.8
s_4	0.5	1.0
s_5	0.5	0.8
s_6	0.5	0.8
s_7	1.0	0.8
s_8	1.0	0.8

$$ER(a_1) = b'''(s_1)R(s_1, a_1) + b'''(s_2)R(s_2, a_1) = 0.2 \cdot 0.5 + 0.8 \cdot 0.5 = 0.5$$

$$ER(a_2) = b'''(s_1)R(s_1, a_2) + b'''(s_2)R(s_2, a_2) = 0.2 \cdot 0.8 + \ 0.8 \cdot 1.0 = 0.96$$

De ahí que el robot se decidirá por la acción:

$$a_2 = GO(X_2)$$

Supongamos que se recibe ahora la observación o_4

Si después de tomar la acción a_2 se recibe la observación o_4

Repetimos el procedimiento

Hay que determinar el nuevo *belief* *b’’’*

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b'''(s')$	0.2	0.8	0.0	0.0	0.0	0.0	0.0	0.0

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
s_1, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_2, a_1	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0
s_3, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_4, a_1	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0
s_5, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_6, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_7, a_1	0.0	0.0	0.2	0.8	0.0	0.0	0.0	0.0
s_8, a_1	0.0	0.0	0.2	0.8	0.0	0.0	0.0	0.0
s_1, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_2, a_2	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5
s_3, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_4, a_2	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5
s_5, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_6, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_7, a_2	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.8
s_8, a_2	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.8

$$P(s'|b''',a_2)=\sum_{s\in S}T(s,a_2,s')b'''(s)$$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s'\|b''', a_2)$	0.0	0.0	0.0	0.0	0.02	0.48	0.02	0.48

Calculando el nuevo *belief (b’’’’)*

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s'\|b''', a_2)$	0.0	0.0	0.0	0.0	0.02	0.48	0.02	0.48

	o_1	o_2	o_3	o_4
$s_1, a_1 \vee a_2$	1.0	0.0	0.0	0.0
$s_2, a_1 \vee a_2$	1.0	0.0	0.0	0.0
$s_3, a_1 \vee a_2$	0.0	1.0	0.0	0.0
$s_4, a_1 \vee a_2$	0.0	1.0	0.0	0.0
$s_5, a_1 \vee a_2$	0.0	0.0	1.0	0.0
$s_6, a_1 \vee a_2$	0.0	0.0	0.0	1.0
$s_7, a_1 \vee a_2$	0.0	0.0	1.0	0.0
$s_8, a_1 \vee a_2$	0.0	0.0	0.0	1.0

$$P(o\,|\,b''',a_2)=\sum_{s'\in S}O(s',a_2,o)P(s'|\,b''',a_2)$$

	o_1	o_2	o_3	o_4
$P(o\|b''', a_2)$	0.0	0.0	0.04	0.96

$$b''''(s')=\frac{O(s',a_2,o_4)P(s'|\,b''',a_2)}{P(o_4\,|\,b''',a_2)}$$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b''''(s')$	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5

Decidir la siguiente acción a tomar

Determinamos la recompensa esperada para las posibles acciones

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b''''(s')$	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5

	a_1	a_2
s_1	0.5	0.8
s_2	0.5	1.0
s_3	0.5	0.8
s_4	0.5	1.0
s_5	0.5	0.8
s_6	0.5	0.8
s_7	1.0	0.8
s_8	1.0	0.8

$$ER(a_1) = b''''(s_6)R(s_6, a_1) + b''''(s_8)R(s_8, a_1) = 0.5 \cdot 0.5 + 0.5 \cdot 1.0 = 0.75$$

$$ER(a_2) = b''''(s_6)R(s_6, a_2) + b''''(s_8)R(s_8, a_2) = 0.5 \cdot 0.8 + 0.5 \cdot 0.8 = 0.8$$

De ahí que el robot se decidirá por la acción:

$$a_2 = GO(X_2)$$

Fin del ejemplo

Belief state	$ER(a_1)$	$ER(a_2)$
b	0.5	**0.8**
b'	0.75	**0.8**
b''	**0.875**	0.8
b'''	0.5	**0.96**
b''''	0.75	**0.8**

A pesar de todas las operaciones involucradas, en realidad solo se tomo un horizonte de 1

Para resolver POMDPs si el horizonte es mayor de 1

La función de actualización del belief state si no se conoce la osbervación, por ejemplo para observaciones futuras

$$\tau(b,a,b') = \Pr(b' \mid a,b) = \sum_{o \in \Omega} \Pr(b' \mid a,b,o) \Pr(o \mid a,b)$$

La función de recompensas sobre estados creídos

$$\rho(b,a) = \sum_{s \in \mathcal{S}} b(s) R(s,a).$$

Aparentemente el agente es recompensado solo por creer que está en un estado bueno, esto no es así, esta función representa la recompensa esperada dado que el belief state tiene la probabilidad de cada estado de ser el estado correcto.

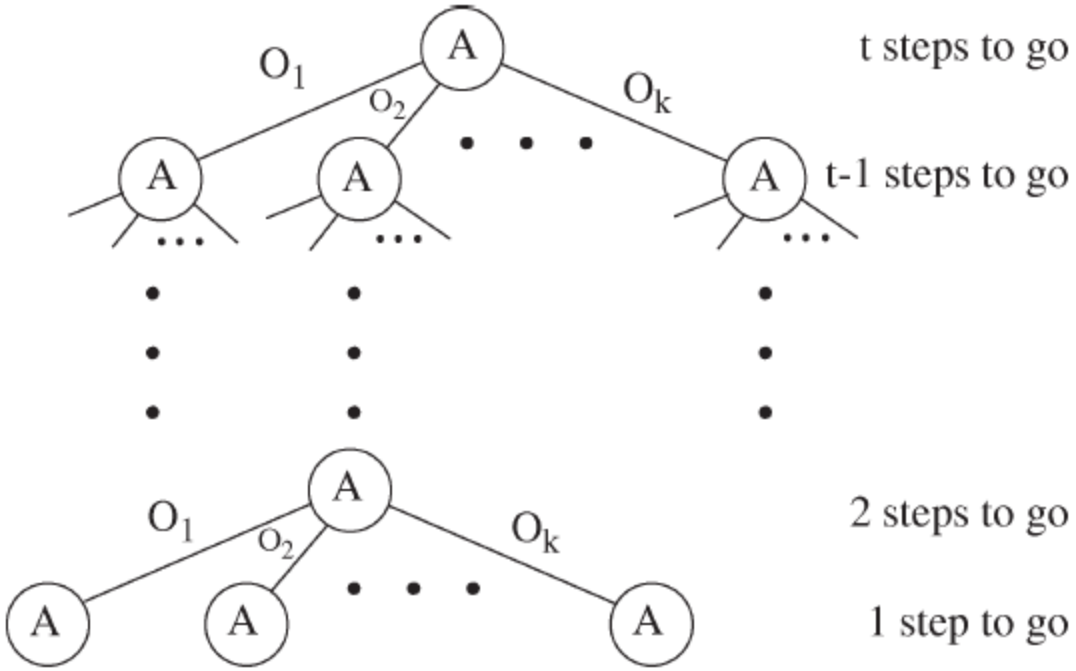
Función de valor óptimo

- Tal como en los MDPs, si podemos determinar la función de valor óptimo, podemos usarla para determinar la política óptima
- Hay varios enfoques para encontrar la función de valor óptimo
- El más simple es el de valor-iteración

Arboles de política

- En el último paso (Ya en el horizonte) simplmente se tomará una acción.
- En el penúltimo paso se tomará una acción, se esperará una observación y luego se tomará otra acción que dependerá de la observación
- Cuando falten t pasos se tomarán una secuencia de decisiones (a lo cual llamamos política)
- Una política no estacionaria se puede representar mediante un árbol de política
- La raíz del árbol indica cual acción se debe tomar primero.
- Dependiendo de la observación se desciende en el árbol para saber cual es la siguiente acción a tomar, etc

Arboles de política



Valor esperado al ejecutar el árbol de política *p*

- Depende del verdadero estado del mundo cuando comienza el proceso
- En el caso más simple, el árbol tiene altura 1 (solo la raíz con una acción), el valor de ejecutar dicha acción

$$V_p(s) = R\big(s, a(p)\big)$$

donde *a(p)* es la acción especificada en la raíz del árbol *p*

Valor esperado al ejecutar el árbol de política *p*

$$
\begin{aligned}
V_p(s) &= R\big(s, a(p)\big) + \gamma \cdot \big(\text{Expected value of the future}\big) \\
&= R\big(s, a(p)\big) + \gamma \sum_{s' \in \mathcal{S}} \Pr\big(s' \mid s, a(p)\big) \sum_{o_i \in \Omega} \Pr\big(o_i \mid s', a(p)\big) V_{o_i(p)}\big(s'\big) \\
&= R\big(s, a(p)\big) + \gamma \sum_{s' \in \mathcal{S}} T\big(s, a(p), s'\big) \sum_{o_i \in \Omega} O\big(s', a(p), o_i\big) V_{o_i(p)}\big(s'\big)
\end{aligned}
$$

$o_i(p)$ es el subárbol de política de *(t-1)* niveles asociado con la observación o_i a la raíz del árbol de política *p*

El valor esperado es la recompensa inmediata esperada cuando se ejecuta la acción especificada en la raíz del árbol mas el valor esperado que se obtendrá considerando todos los posibles estados calculando por cada estado el valor esperado tomando en cuenta todas las posibles observaciones

El valor óptimo de un *belief*

- Como el agente no conoce el verdadero estado del mundo, deberá determinar el valor esperado al ejecutar la política p teniendo para ello solo el belief state $V_p(b) = \sum_{s \in \mathcal{S}} b(s) V_p(s)$

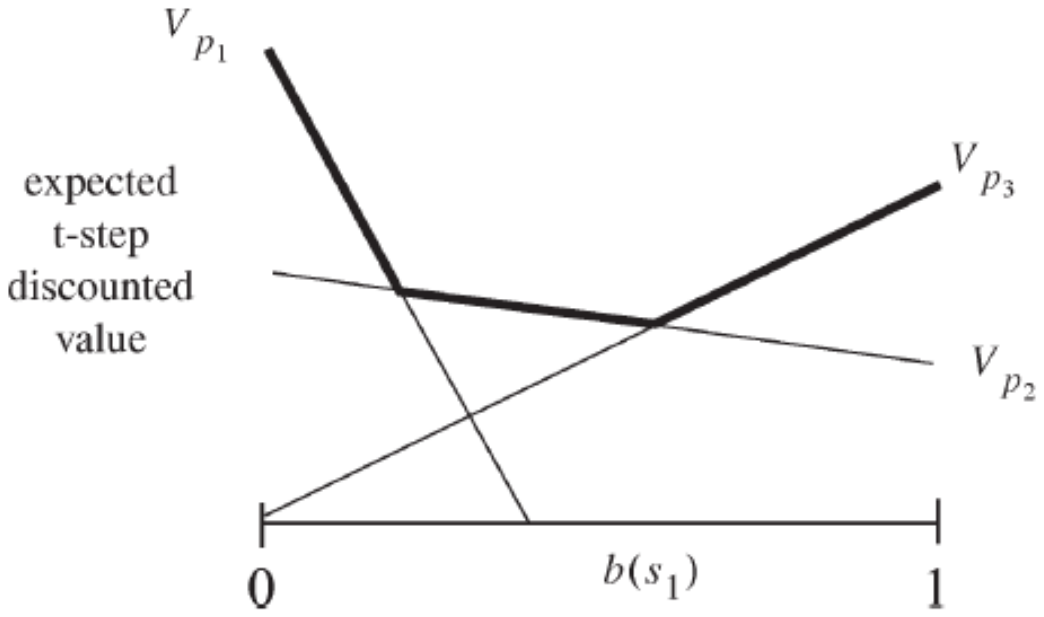
En forma compacta, si $\alpha_p = \langle V_p(s_1), \ldots, V_p(s_n) \rangle$

entonces $V_p(b) = b \cdot \alpha_p$

- Para encontrar una política óptima no estacionaria debemos ser capaces de ejecutar diferentes árboles de política desde diferentes beliefs $V_t(b) = \max_{p \in \mathcal{P}} b \cdot \alpha_p$ $\mathcal{P}$ es el conjunto finito de árboles de política de t pasos
- El valor óptimo correspondiente *b* es el valor de ejecutar el mejor árbol de política desde *b*

PWLC

- A cada árbol de política *p* le corresponde una función de valor *Vp* que es lineal en *b*
- *Vt* es el techo de dicha colección de funciones
- Por tanto *Vt* es PWLC (Piecewise linear and convex)

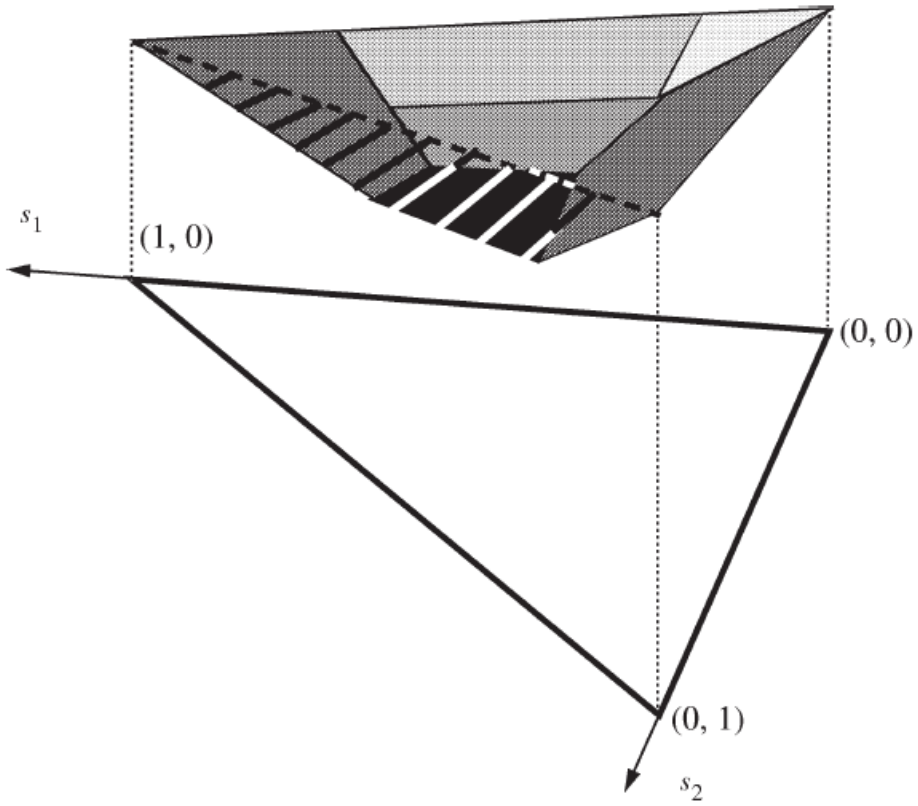


En este ejemplo solo hay dos estados, *b* es un vector de solo dos elementos positivos que suman uno, por tanto se puede representar por uno solo de ellos

Función valor para 3 estados

- Para 3 Edos, b es un vector de 3 elems positivos que suman 1 por lo que se puede representar por solo dos de ellos
- El belief space es un triángulo con vértices (0,0), (1,0) y(0,1)
- El valor, asociado con un árbol del política específico es un plano en 3D
- La función de valor óptimo tiene forma de un tazón hecho con muchas caras planas

Para mayor número de estados, el valor asociado a cada política es un hiperplano y la función valor tiene el mayor valor de todos los hiperplanos



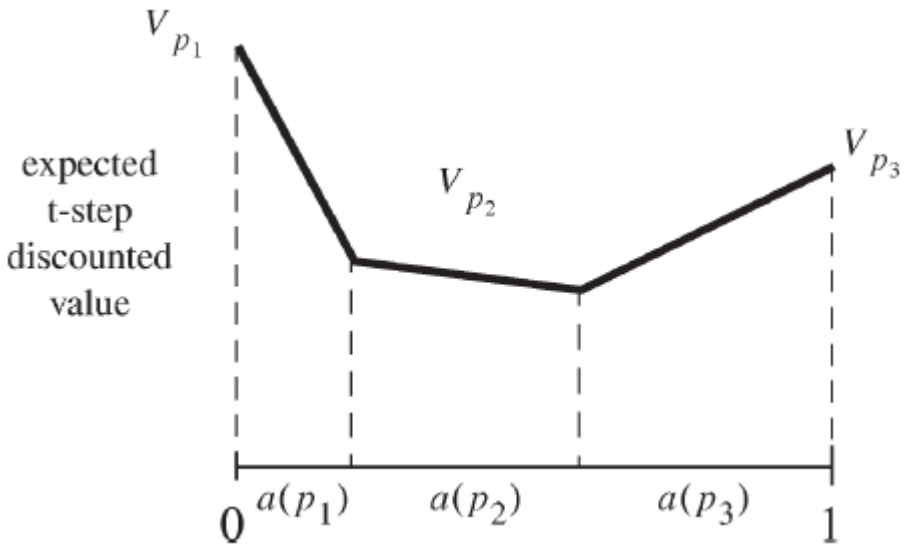
La función valor es convexa

- En los extremos del espacio *b* el agente tiene gran confianza respecto al estado real en que se encuentra el mundo y puede tomar decisiones buenas que le retribuyen mucho
- En la parte central del espacio *b*, el agente está muy confundido acerca del estado real del mundo y no suele tomar decisiones adecuadas
- *b* es una distribución por lo que sabemos si estamos en el centro del espacio *b* calculando su entropía
- Alta entropía significa que estamos en una parte del centro y el agente tiene poca información
- Baja entropía significa que estamos cerca de un extremo y el agente tiene mucha información
- En lugares donde el agente tiene poca información debe pagar un costo para moverse a lugares donde tendrá mas información
- El agente paga por el *valor de la información*
- Solo debe pagar si la diferencia de valor entre los dos estados excede al valor de la información

Regiones de decisión

- Dada una función de valor óptimo (sabemos que es PWLC) y el conjunto de árboles de política de donde se obtuvo, podemos hacer un mapeo situación-acción
- Proyectamos la función valor óptimo en el espacio de *b* (de donde originalmente se obtuvo)
- Esto produce una partición del espacio *b* en regiones poliédricas
- Dentro de cada región hay un solo árbol de política *p* tal que $b \cdot \alpha_p$ es óptimo para toda la región y la acción óptima es *a(p)*

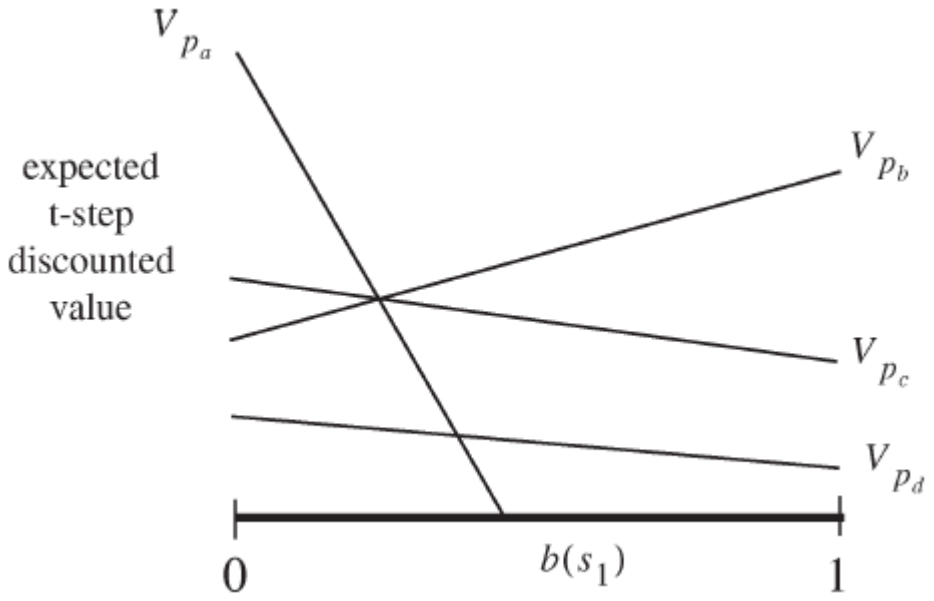
Incluso se podría ejecutar el árbol de decisión completo tomando decisiones solamente basándose en las futuras observaciones sin necesidad de actualizar el belief



Algunos árboles pueden ignorarse

- Es posible que cada uno de los árboles de política contribuya en algún punto del espacio de b a la determinación de la función de valor óptimo
- Afortunadamente ese no es el caso, para muchos árboles, sus funciones de valor son totalmente dominadas por funciones de valor asociadas a otros árboles
- *Pd* está totalmente dominado por *Pb* y por *Pc*
- *Pc* está totalmente dominado por *Pa* y *Pb* combinados

- Dado un conjunto de árboles se puede obtener un subconjunto mínimo que represente la misma función de valor óptimo.
- Decimos que un árbol es útil si es miembro de este conjunto mínimo



Ejemplo, El problema del tigre

- Un agente tiene dos puertas cerradas frente a sí
- Detrás de una puerta hay un tigre
- Detrás de la otra puerta hay una recompensa enorme
- Si abre la puerta del tigre recibirá un fuerte castigo (Ej perder un brazo)
- En lugar de abrir una puerta el agente puede escuchar para ganar información
- Escuchar no es gratis
- Escuchar no es tan confiable, existe la posibilidad de escuchar al tigre en la puerta izquierda cuando el tigre realmente está en la puerta derecha y vicebersa

El problema del tigre

- Nos referiremos al verdadero estado del mundo cuando el tigre está en la puerta izquierda como *sl* y si está en la puerta derecha como sr
- Las acciónes son: LEFT, RIGHT, y LISTEN
- La recompensa por abrir la puerta correcta es de +10 y de -100 por abrir la puerta erronea
- El costo por escuchar es -1
- Hay dos observaciones: TL es escuchar el tigre a la izquierda y TR es escuchar al tigre a la derecha
- Inmediatamente después de abrir una puerta y recibir por ello un castigo o una recompensa el tigre se reubica aleatoriamente detrás de alguna de las dos puertas

El problema del tigre

- La acción LISTEN no cambia el verdadero estado del mundo
- Las acciones LEFT y RIGHT provocan una transición al estado *sl* con probabilidad 0.5 y al estado *sr* con probabilidad 0.5
- Cuando el tigre está en la puerta izquierda, la acción LISTEN resulta en la observación TL con probabilidad 0.85 y en la observación TR con probabilidad 0.15
- Cuando el tigre está en la puerta derecha, la acción LISTEN resulta en la observación TL con probabilidad 0.15 y en la observación TR con probabilidad 0.85
- Sin importar el verdadero estado del mundo, las acciones LEFT y RIGHT provocan la observación TR con probabilidad 0.5 y la observación TL con probabilidad 0.5

Para t=1

- Para t=1 el agente solo tiene que tomar una decisión.
- Si el agente cree que es muy probable que el tigre esté detrás de la puerta izquierda, la mejor acción será RIGHT (abrir la puerta derecha)
- Si cree que es muy probable que esté detrás de la puerta derecha abrirá la puerta izquierda (LEFT)
- Si el agente tiene demasiada incertidumbre acerca de la ubicación del tigre la mejor acción será LISTEN
- Se muestran 3 árboles de política con altura de 1 (un solo nodo), la acción y el intervalo de b donde la política predomina
- Por ejemplo si el b=(0.5 0.5), la recompensa esperada es de

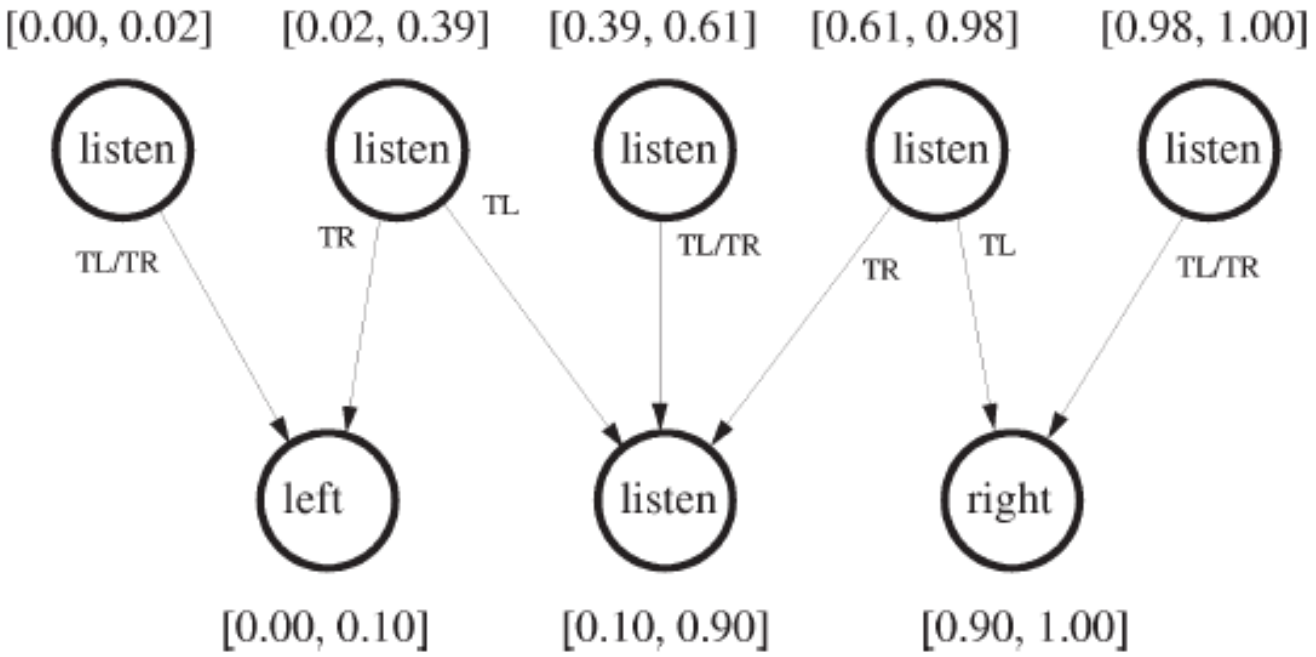
(-100+10)/2=-45 y como escuchar vale -1 que es mayor entonces prefiere escuchar

t=2

- La formar los árboles de política para t=2 comenzamos por el siguiente mapeo situación acción
- Sorprendentemente siempre opta por escuchar, esto se debe a que si abre una puerta en el siguiente paso el tigre se reubicará aleatoriamente detrás de cualquiera de las dos puertas, por tanto, después de abrir una puerta se quedará sin información para el siguiente paso y como vimos con *b*=(0.5,0.5) lo mejor es escuchar
- Si escucha estará mejor informado para tomar una decisión en el siguiente paso
- Hay muchas políticas con la misma acción por que hay muchos árboles de política diferentes pero que tienen en la raíz la misma acción
- Esto implica que la función valor no es lineal sino que está formada de 5 regiones lineales

Relación entre las regiones de decisión para t=2 y para t=1

- Cuando un *b* especifico dentro de alguna de las regiones de decisión para *t*=2 es actualizado vía SE(*b,a,o*), el *b'* resultante cae dentro de alguna de las regiones de decisión para *t*=1

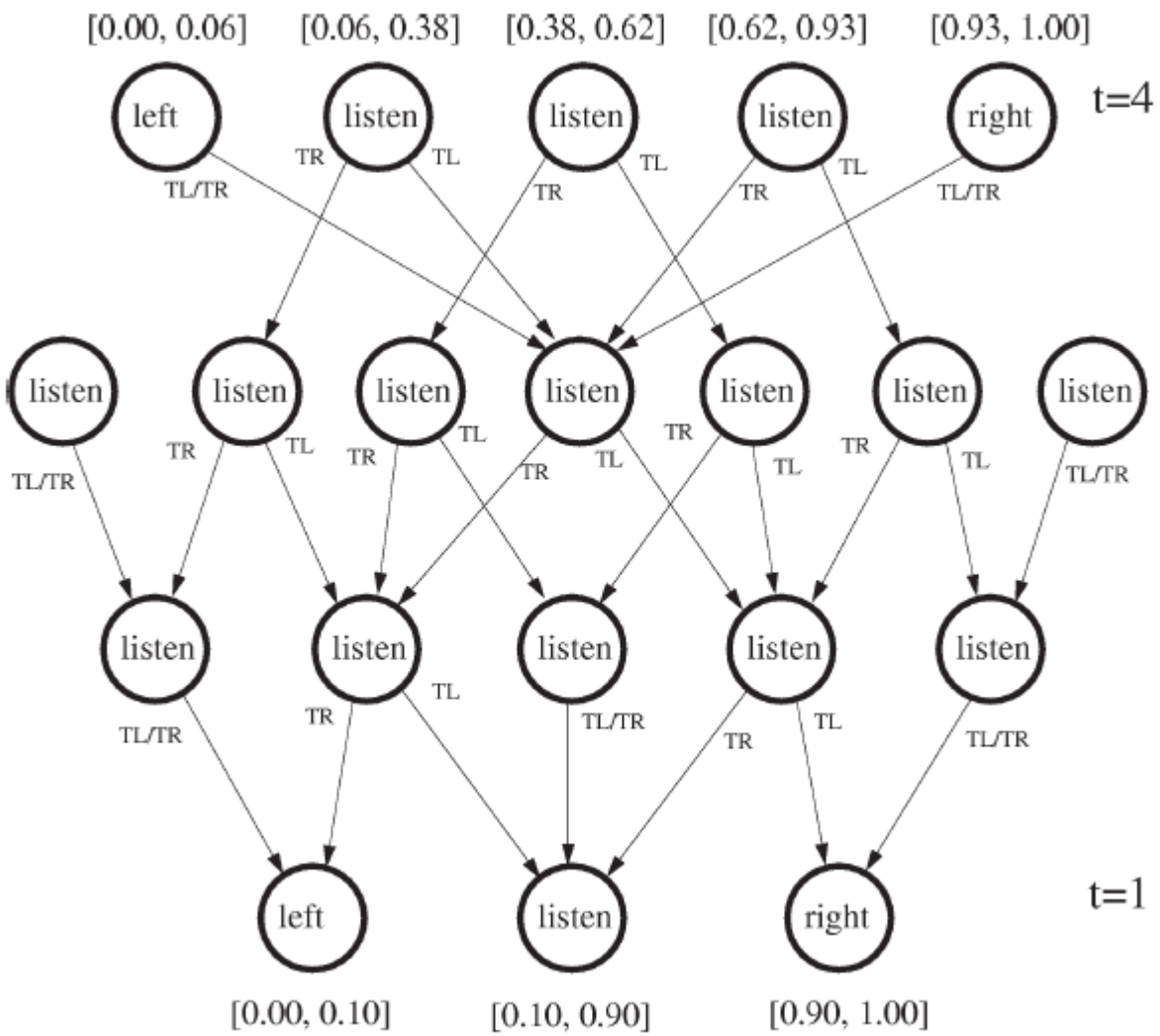


t=3

- Las políticas no estacionarias para t=3 están formadas solamente de árboles de política que tienen en la raíz a la acción LISTEN
- Esto se debe a que escuchar una sola vez no cambia lo suficiente el belief state para hacer que la recompensa esperada al abrir una puerta sea mayor que la de escuchar.

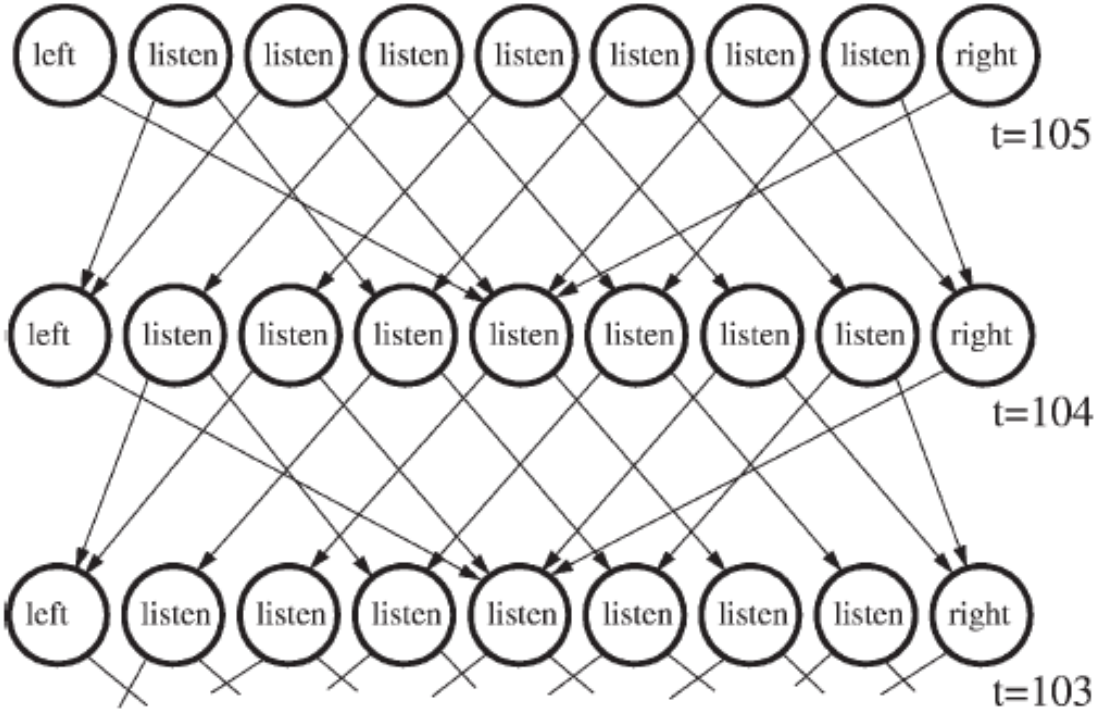
t=4

- Algunos nodos de la tercera etapa no tienen flechas dirigidas hacia ellos porque ningún belief de ninguna región de decisión de la etapa anterior podría ser actualizado para convertirse en un belief que corresponda a la región de decisión correspondiente a esos nodos
- Este grafo puede interpretarse como una representación compacta de todos los árboles de política útiles para cada nivel



Horizonte infinito

- Al incluir un factor de descuento, la estructura de la función valor cambia ligeramente
- A medida que el horizonte t aumenta la recompensa recibida en los último pases tiene menor influencia en como se hace el mapeo situación-acción de los pasos iniciales y la función valor comienza a converger
- Las regiones de decisión de una etapa a la siguiente difieren muy poco a partir de t=56. A partir de t=105 la precisión no permite diferencia una etapa de la que sigue

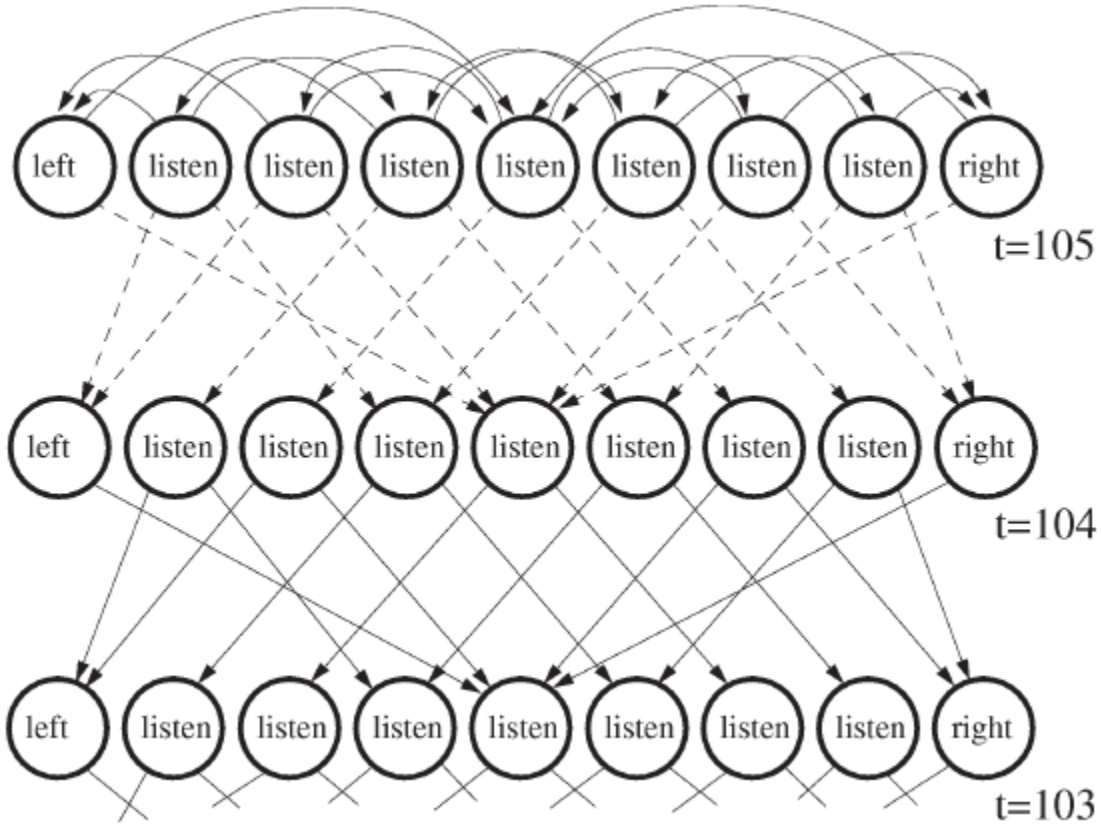


Grafos de planes

- Una desventaja del enfoque del POMDP es que el agente debe mantener el belief state y usarlo para elegir la acción óptima en cada paso
- Si el espacio de estados es grande, estos cómputos pueden ser excesivamente costosos
- En muchos casos es posible codificar la política en un grafo que puede ser usado para elegir acciones sin necesidad de una representación explícita del belief state
- Estos grafos se denominan “Grafos plan”

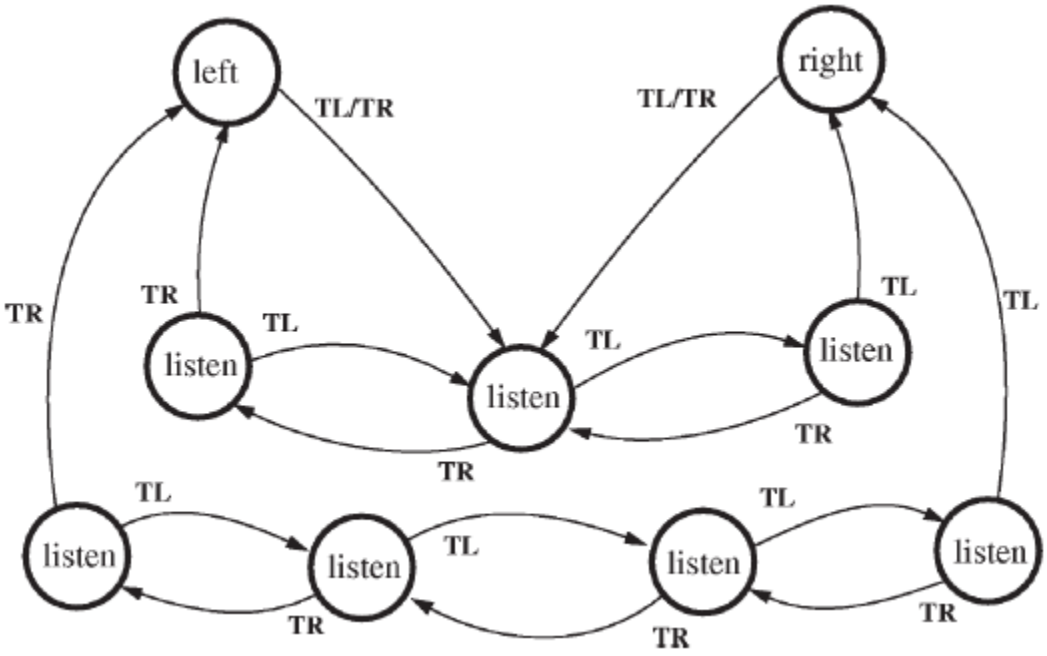
Grafo plan para el problema del tigre con horizonte infinito

- Como el mapeo situación-acción tiene la misma estructura para las diferentes etapas se puede fácilmente convertir la política dinámica en una política estática



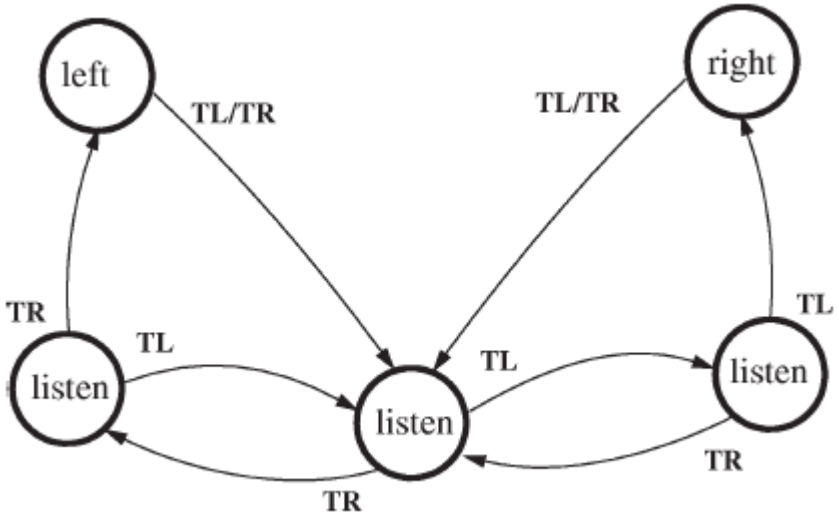
Mejorando en grafo plan

- Algunos nodos no serán revisitados una vez que una puerta se ha abierto pues el belief regresa a (0.5,0.5)
- La interpretación de este grafo es “Sigue escuchando hasta que hayas escuchado el doble de veces que el tigre se encuentra de un lado respecto al numero de veces que has escuchado que se encuentra del otro lado

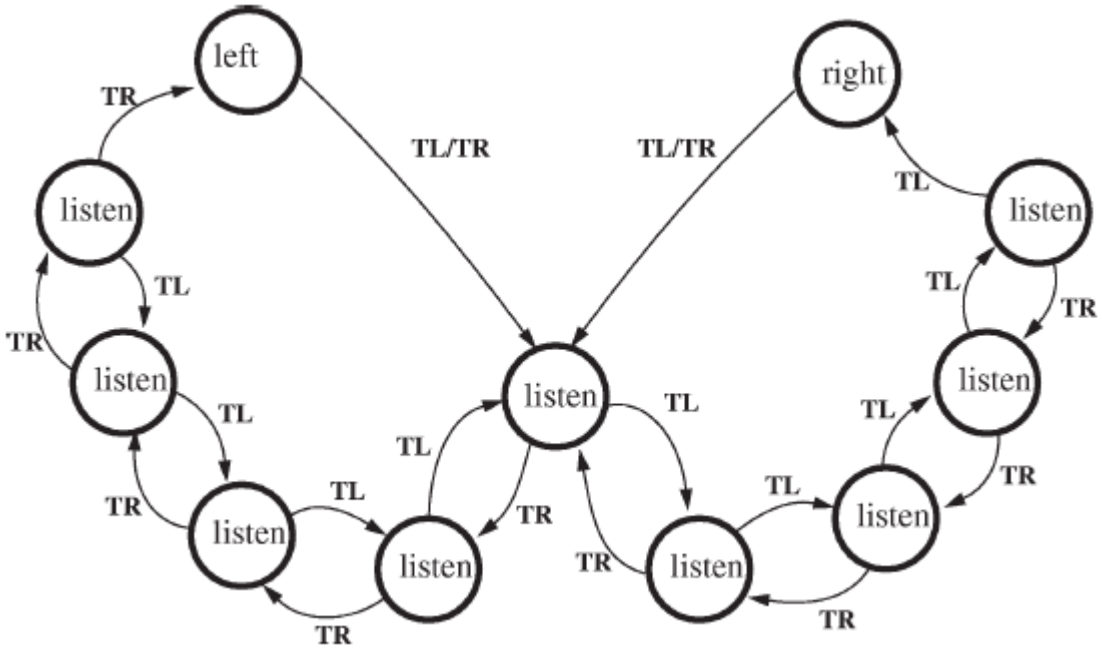


Mejorando el grafo plan

- Dado que el belief inicial es (0.5,0.5) se pueden eliminar algunos nodos del grafo plan



Mas requerimiento de memoria si la credibilidad de la observación es 0.65



- El agente debe escuchar 5 veces al tigre de un lado respecto al número de veces que lo escuchó del otro lado

Notas finales respecto al grafo plan

- Un grafo plan es esencialmente el controlador de una máquina de estados finita
- Utiliza muy poca memoria para actuar de manera óptima en un ambiente parcialmente observable
- Comenzamos con un problema discreto, lo reformulamos en términos de un espacio belief continuo y al final lo mapeamos de regreso en un controlador discreto
- La determinación del controlador se puede automatizar a partir de dos funciones valor sucesivas que sean iguales