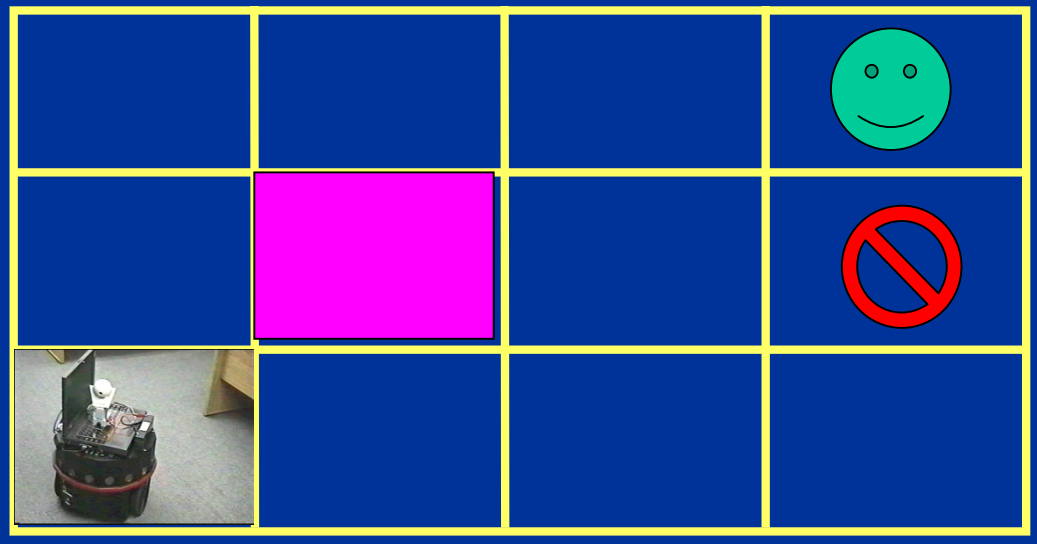
Modelos Gráficos Probabilistas

L. Enrique Sucar

INAOE

Sesión 15: Procesos de Decisión de Markov



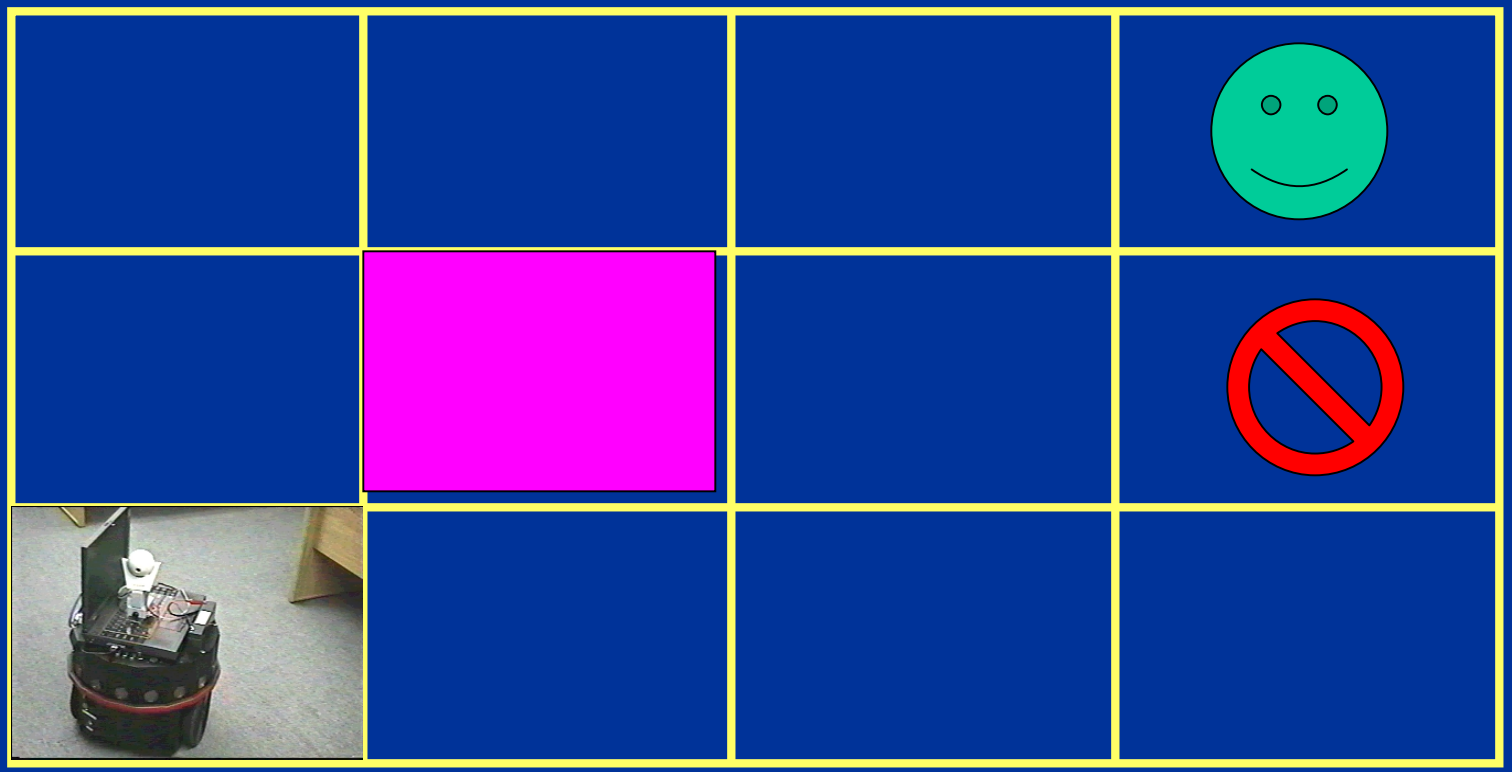
Procesos de Decisión de Markov

- Procesos de Decisión Secuenciales
- Procesos de Decisión de Markov (MDPs)
- Técnicas de Solución:
  - Iteración de Valor
  - Iteración de Política
- MDPs Parcialmente Observables (POMDPs)
- Extensiones
  - MDPs factorizados
  - Abstracción, descomposición
- Aplicaciones

Problemas de decisión secuenciales

- Problema de decisión que involucra un conjunto de decisiones cuyo resultado (utilidad) se conoce hasta el final
- Se considera que se tiene una serie de estados y decisiones asociadas en el tiempo
- Se tiene incertidumbre asociada con los resultados de las acciones (MDP), y posiblemente también con los estados (POMDP)

Ejemplo – robot móvil



Inicio

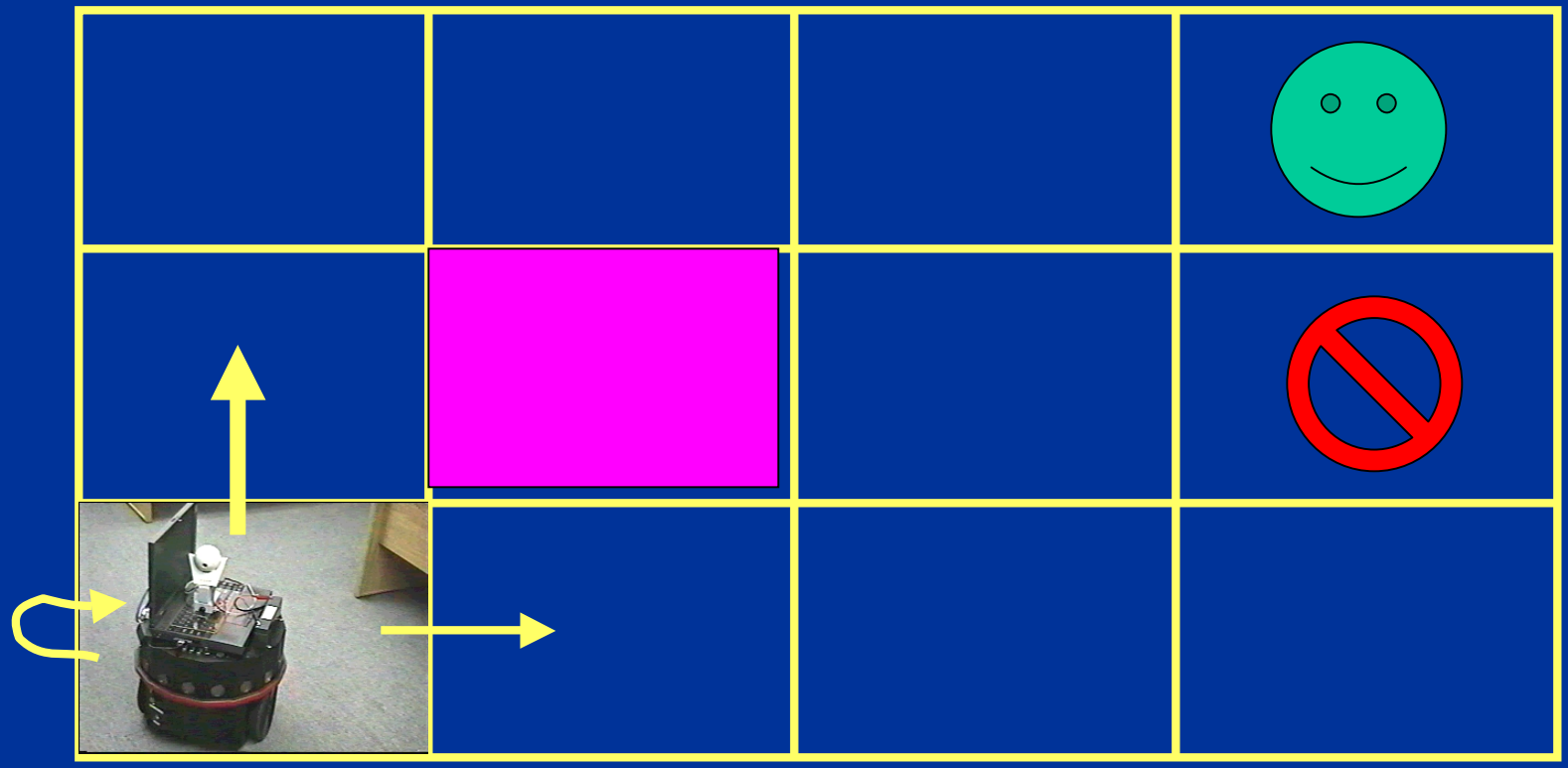
Modelo de Transición

- Normalmente existe incertidumbre respecto a los resultados de una decisión (acción)
- Esta incertidumbre se modela como una probabilidad de llegar al estado “j” dado que se encuentra en el estado “i” y se realizá la acción “a”:

$$P_{ij}^{a}$$

Modelo de Transición

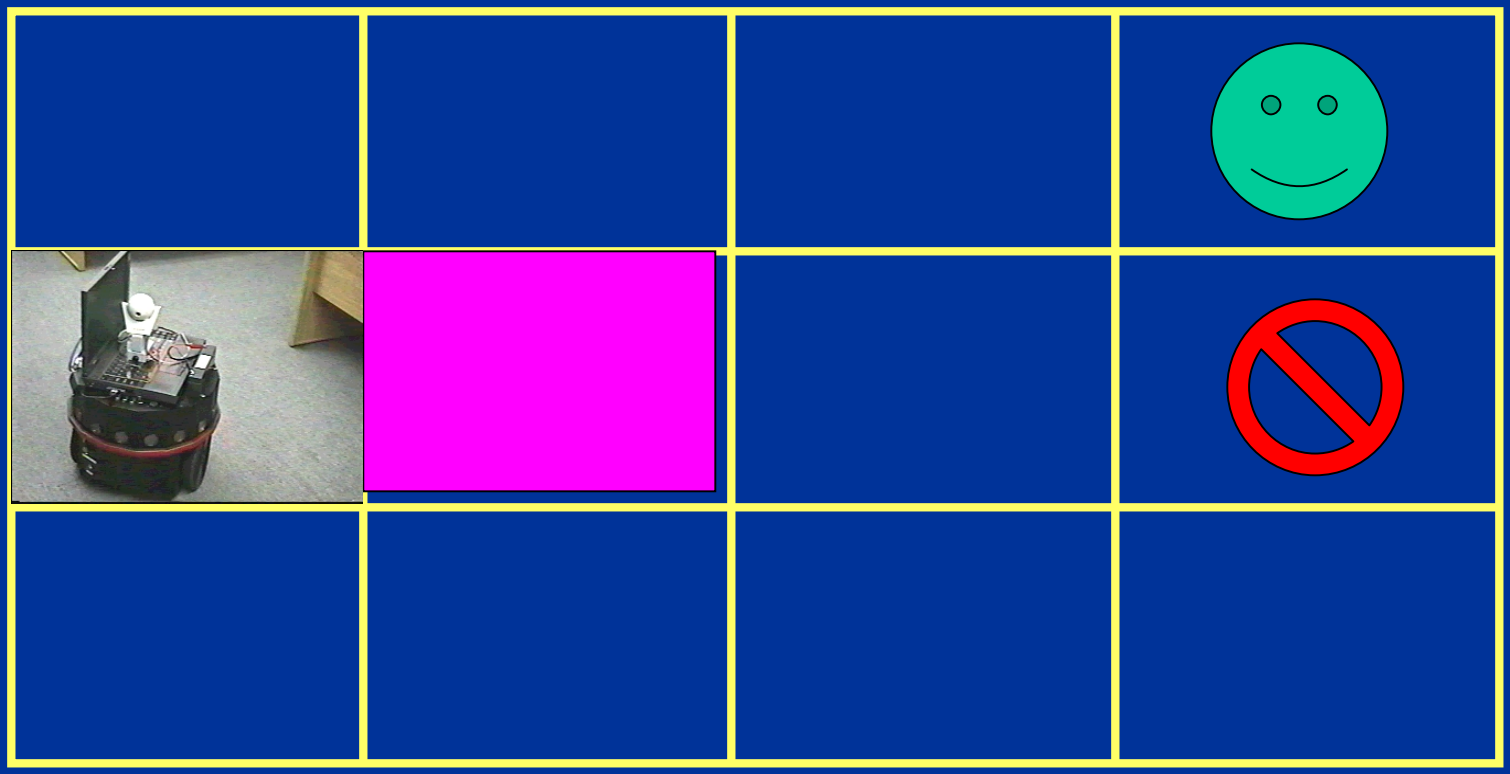
- Probabilidad dirección deseada – Pij=0.8
- Probabilidad 2 direcciones vecinas – Pik=0.1



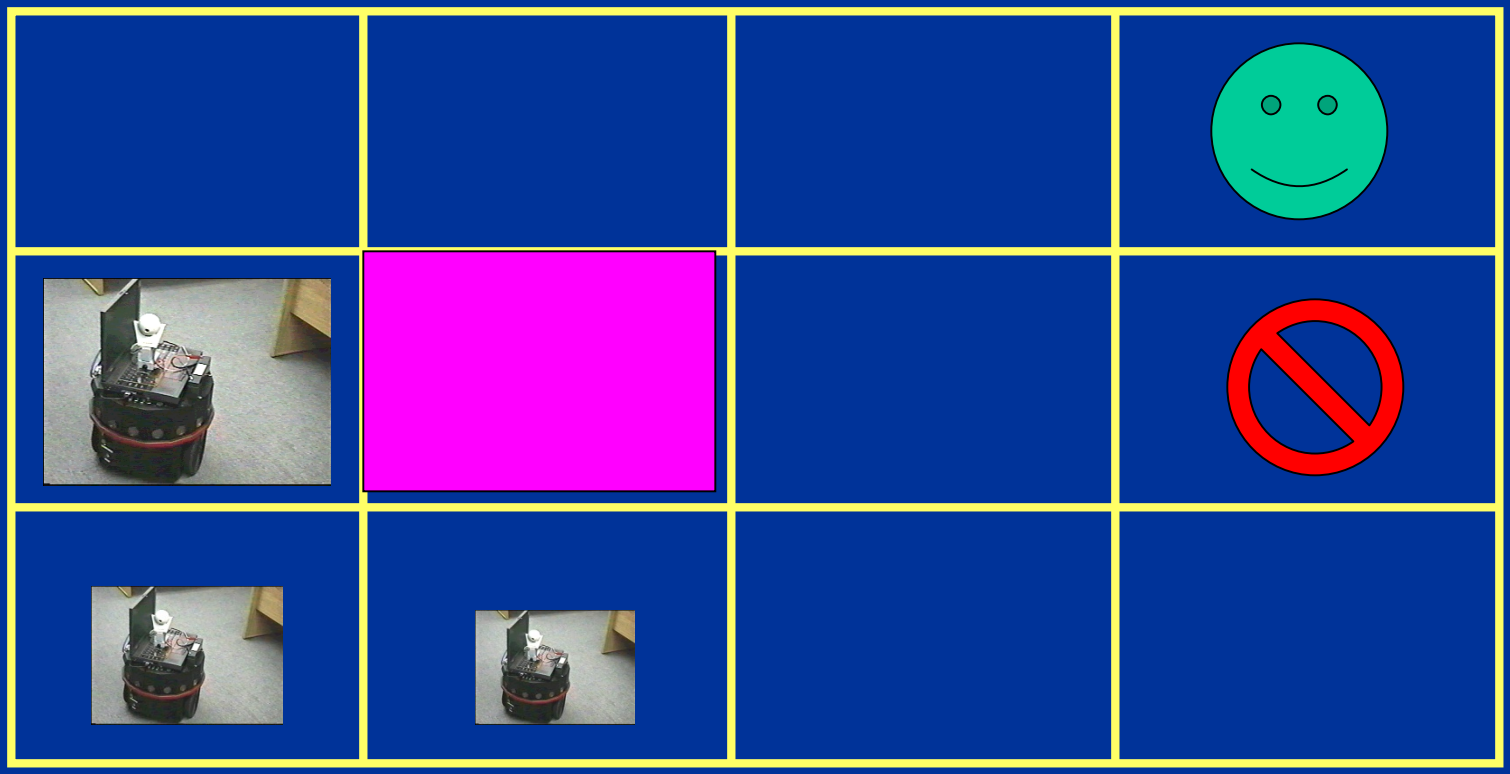
Modelo de los Sensores

- Normalmente el agente puede sensar el ambiente para observar en que estado se encuentra.
- Existen dos casos principales:
  - Observa directamente el estado donde se encuentra- proceso de decisión de Markov
  - Se tiene incertidumbre sobre el estado en que se encuentra- proceso de decisión de Markov parcialmente observable

MDP



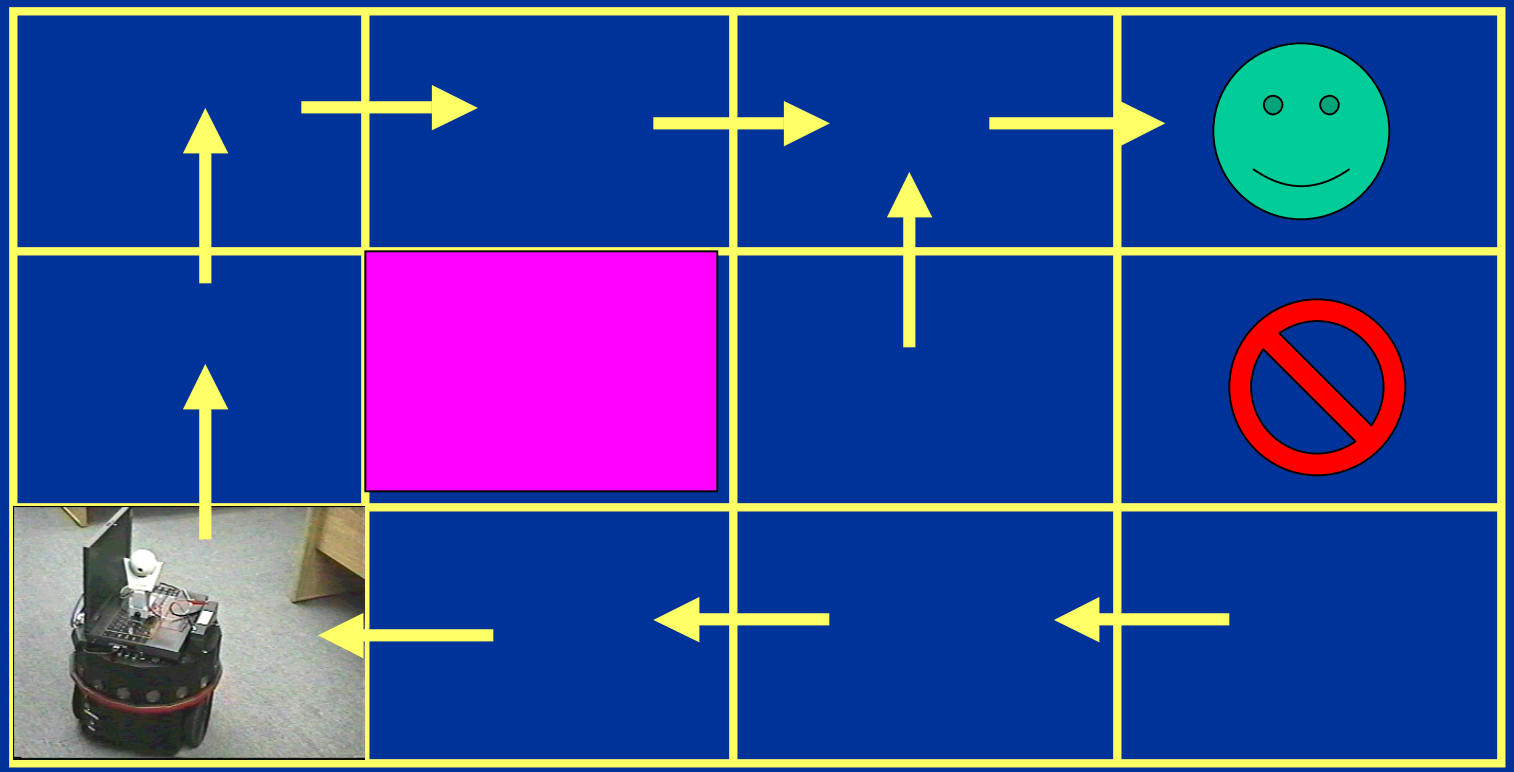
POMDP



Política Óptima

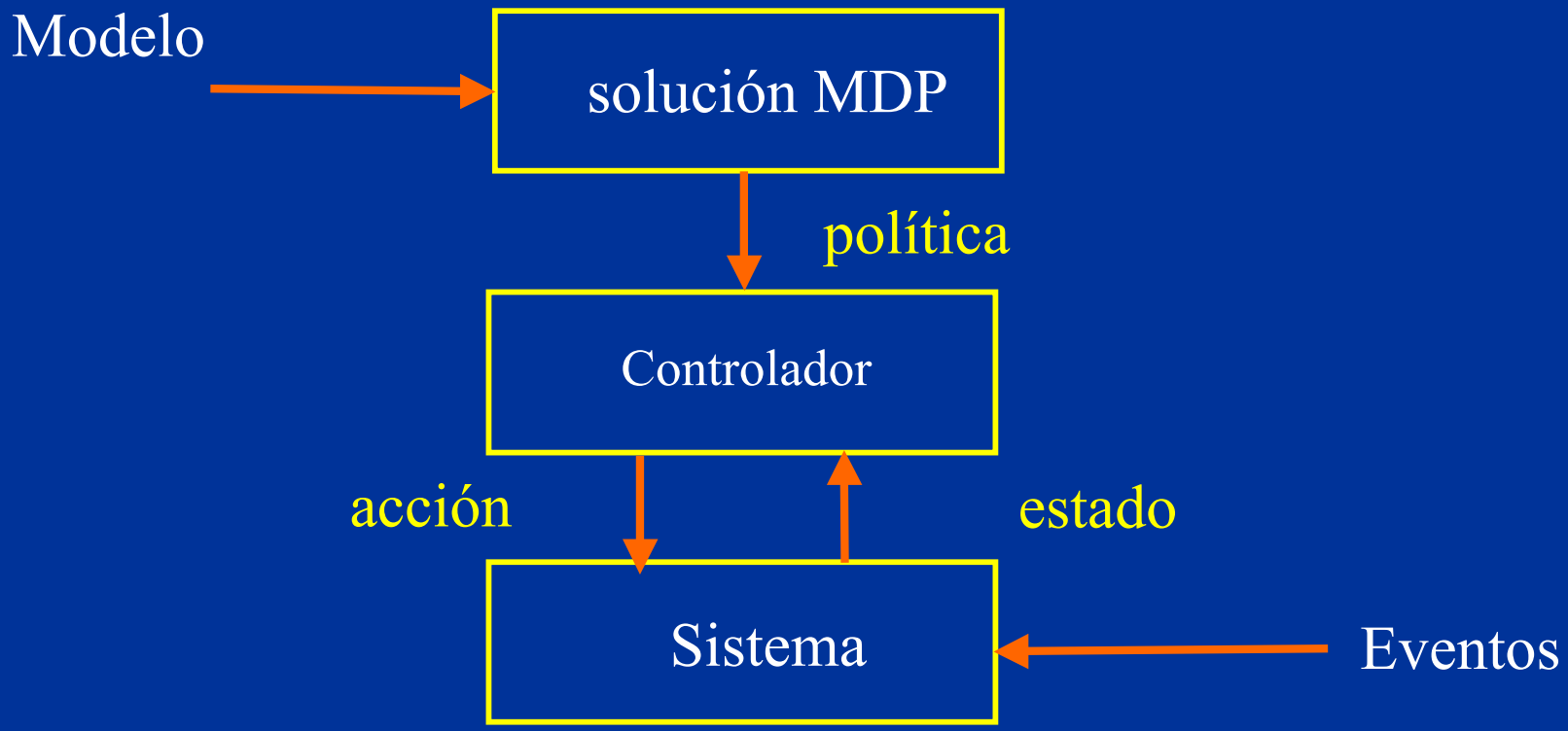
- Dado el modelo de transición y el modelo de los sensores, el objetivo es encontrar la *política óptima* para maximizar la utilidad esperada
- Una política indica la acción que se debe ejecutar dado el estado (o probabilidad del estado)
- Se considera que las probabilidades de transición sólo dependen del estado actual por lo que son *procesos markovianos*

Ejemplo de Política



Inicio

Controlador basado en un MDP



Procesos de Decisión de Markov

- Problema de obtener la política óptima en un ambiente observable – MDP
- El método clásico para resolver estos problemas se conoce como "iteración de valor" (*value iteration*)
- La idea básica es calcular la utilidad de cada posible estado y usar éstas para seleccionar la acción óptima en cada estado
- Otros métodos de solución son "iteración de política" (*policy iteration*) y programación lineal (al transformar el problema a un problema de optimización lineal)

Procesos de Decisión de Markov

- Formalmente, un MDP (discreto) se define por:
  - Un conjunto finito de estados, S
  - Un conjunto finito de posibles acciones, A
  - Un modelo de transición, que especifica la probabilidad de pasar a un estado dado el estado presente y la acción, P(s | s', a)
  - Una función de recompensa, que especifica el "valor" de ejecutar cierta acción *a* en el estado *s*, r(s, a)

Utilidad

- La utilidad de un estado depende de la secuencia de acciones tomadas a partir de dicho estado (i) de acuerdo a la política establecida (p)
- En principio, se puede obtener como la utilidad esperada de todas las posibles secuencias de acciones (Hi) y la utilidad resultante para c/u:

$$U(i) = UE(\ Hi(p)\ ) = \Sigma\ P(Hi(p))\ U_h\ Hi(p)$$

Utilidad

- Si la utilidad es separable, se puede estimar como la utilidad del estado presente y la utilidad de los siguiente estados
- La forma más sencilla es que sea una función aditiva:

$$U[s_0, s_1, \ldots s_n] = R(s_0) + U[s_1, \ldots s_n]$$

- Donde *R* se conoce como la función de recompensa

Programación Dinámica

- Dada la condición de separabilidad, la utilidad de un estado se puede obtener en forma iterativa maximizando la utilidad del siguiente estado:

$$U(i) = R(i) + \max_a \sum_j P(s_j \mid s_i,a)\, U(j)$$

- La política óptima esta dada por la acción que de mayor utilidad:

$$P^*(i) = \arg\max_a \sum_j P(s_j \mid s_i,a)\, U(j)$$

Horizonte finito vs. infinito

- Los problemas con un número finito de pasos se conocen como MDP de horizonte finito
- Los problemas en que puede haber un número infinito de pasos se conocen como MDP de horizonte infinito
- Muchos problemas, como el ejemplo del robot, son de horizonte infinito

Solución

- Los métodos principales para resolver MDPs son:
  - Iteración de valor (Bellman, 57),
  - Iteración de política (Howards, 60),
  - Programación lineal (Puterman, 94).

MDPs – ecuaciones fundamentales

- Función de valor (ecuación de Bellman):

$$V^*(s) = \max_a \{ R(s,a) + \gamma \sum_{s'} P(s' \mid s, a) V^*(s') \}$$

- Política:

$$\pi^*(s) = \arg\max_a \{ R(s,a) + \gamma \sum_{s'} P(s' \mid s, a) V^*(s') \}$$

Donde γ es un factor de descuento

Solución

Función de valor

- Una política para un MDP es una asociación π:S $\rightarrow$A (acción por estado).
- Dada la política, el valor para horizonte finito es: V_n^{π}: S $\rightarrow \Re$

$$V_n^{\pi}(i) = R(i, \pi(i)) + \Sigma P(\pi(i) \mid i,j)\, V_{n-1}(j)$$

- Para horizonte infinito, generalmente se considera un *factor de descuento*, $0<=\gamma<1$:

$$V^{\pi}(i) = R(i, \pi(i)) + \gamma\Sigma P(\pi(i) \mid i,j)\, V(j)$$

Solución

Política óptima

- La solución a un MDP da una política óptima.
- Esto es, la política que maximiza la ecuación de Bellman:

$$\pi^*(i) = \max \left[R(i, a) + \gamma \sum P(a \mid i,j)\, V^*(j) \right]$$

Iteración de Valor

- En el caso de horizonte infinito, se puede obtener la utilidad de los estados –y la política óptima, mediante un método iterativo
- En cada iteración (t+1), se estima la utilidad de cada estado basada en los valores de la iteración anterior (t):

$$U_{t+1}(i) = R(i) + \max_a \sum_j P(s_j \mid s_i,a)\, U_t(j)$$

- Cuando $t \rightarrow \inf$, los valores de utilidad convergen a un valor estable

Iteración de Valor

Algoritmo:

- Inicializar: $U_t = U_{t+1} = R$
- Repetir:
  - $U_t = U_{t+1}$
  - $U_{t+1}(i) = R(i) + \max_a \sum_j P(s_j \mid s_i, a)\, U_t(j)$
- Hasta: $| U_t - U_{t+1} | < \varepsilon$

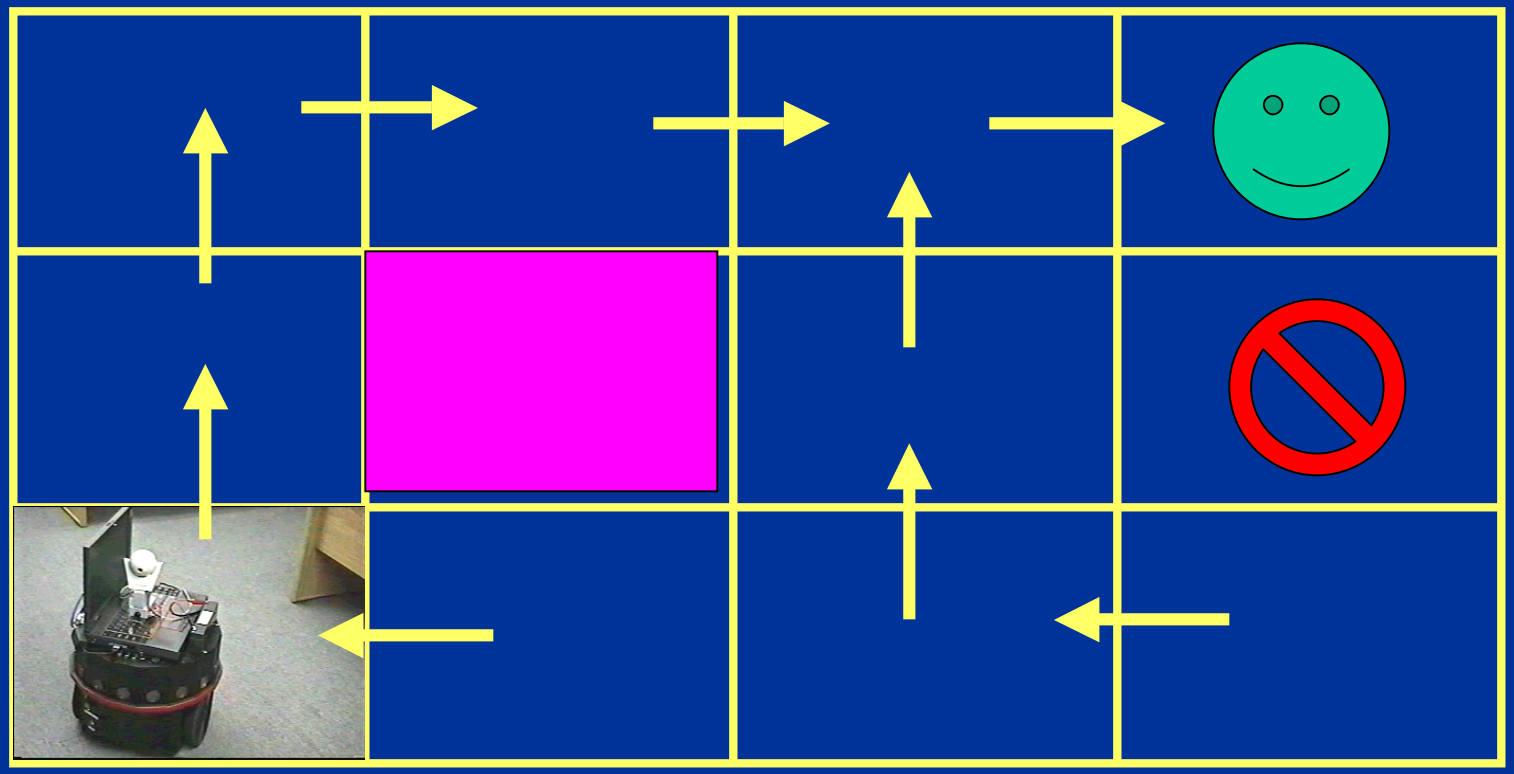
Iteración de Valor

- ¿Cuántas veces repetir la iteración?
- Normalmente el número de iteraciones para obtener la política óptima es menor que el requerido para que las utilidades converjan
- En la práctica, el número de iteraciones es relativamente *pequeño*

Ejemplo – utilidades de los estados

0.812	0.868	0.912	
0.762		0.660	
0.705	0.655	0.611	0.338

Ejemplo – política óptima



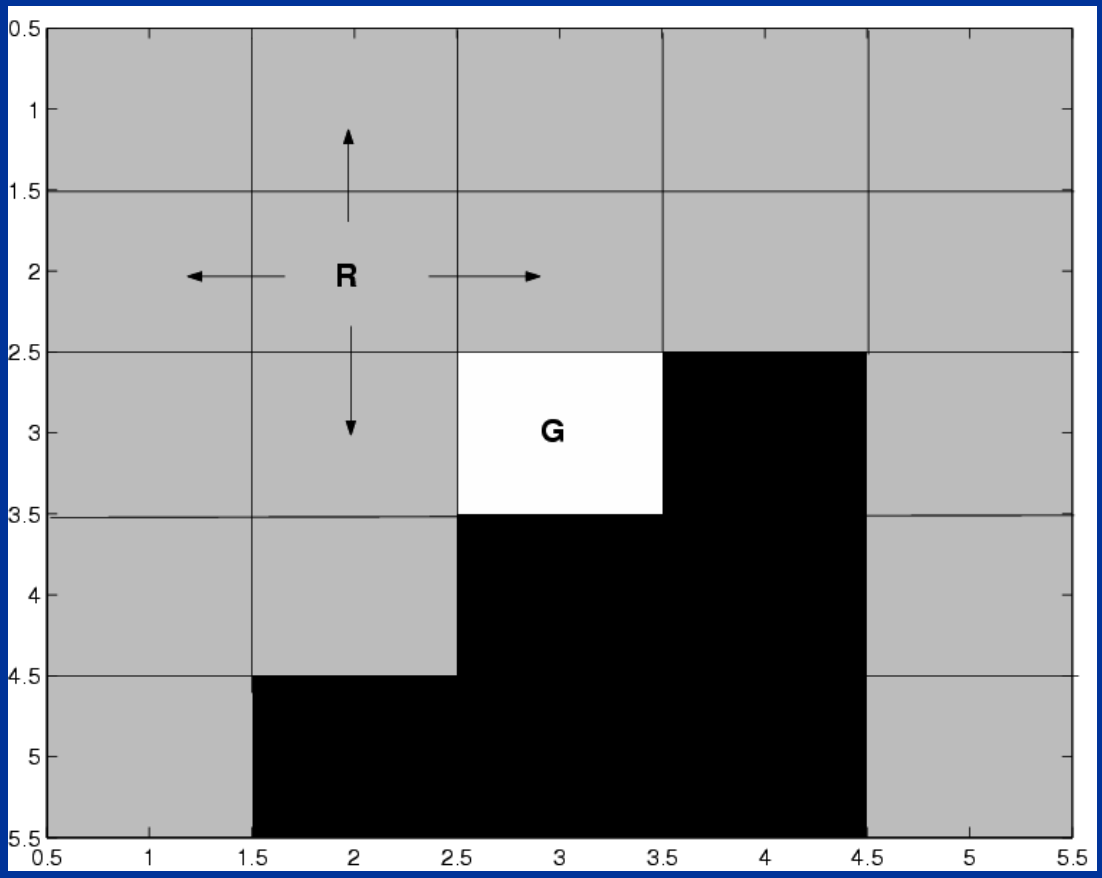
Iteración de Política

- Empezando de cierta política (aleatoria), esta se mejora encontrando una acción por estado que tenga un mejor valor que la acción actual
- Se puede usar conocimiento del problema para definir la política inicial
- El proceso termina cuando ya no puede haber mejoras
- Normalmente converge en menor número de iteraciones que iteración de valor, pero cada iteración es más costosa

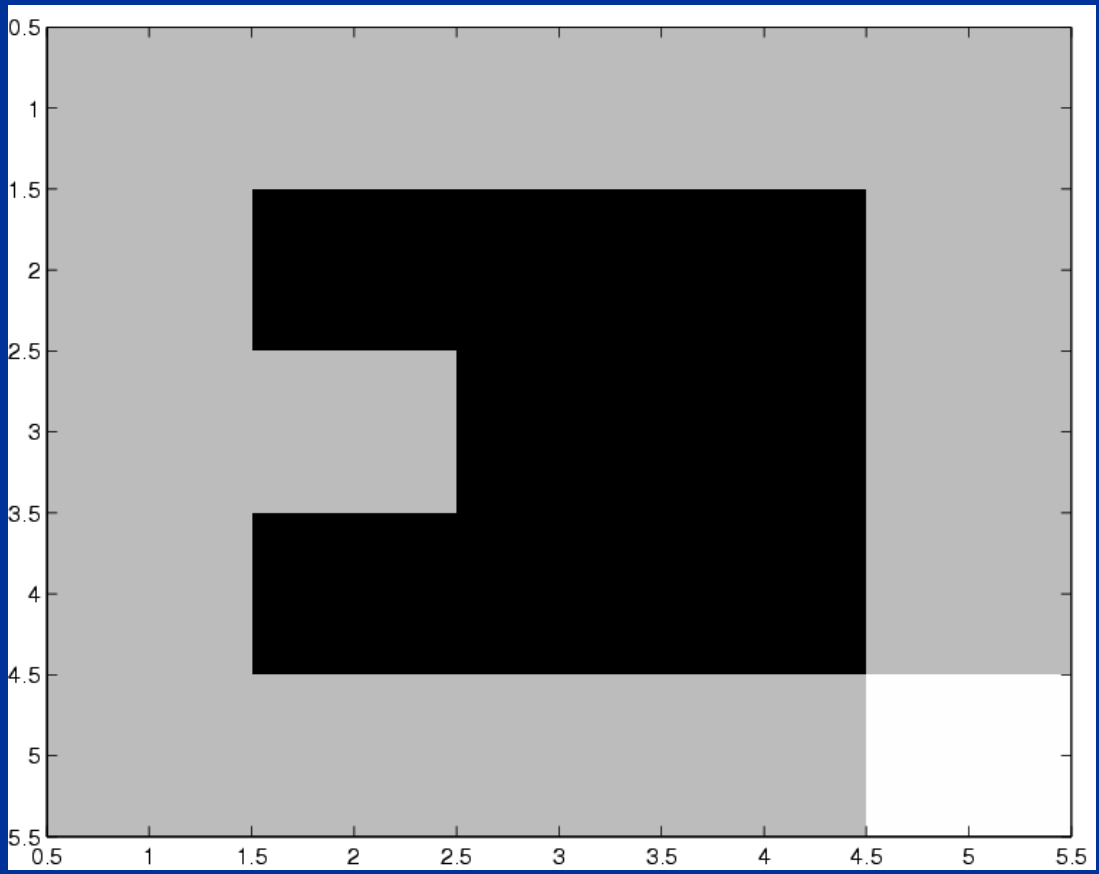
Iteración de Política

- Escoger una política inicial
- Repetir hasta convergencia:
  - Obtener el valor para todos los estados, V^{π}, basado en la política actual π
  - Para cada acción, *a*, calcular: $Q_a = R + \gamma P_a V^{\pi}$
  - Redefinir: $\pi(s) = \text{argmax}_a Q_a(s)$

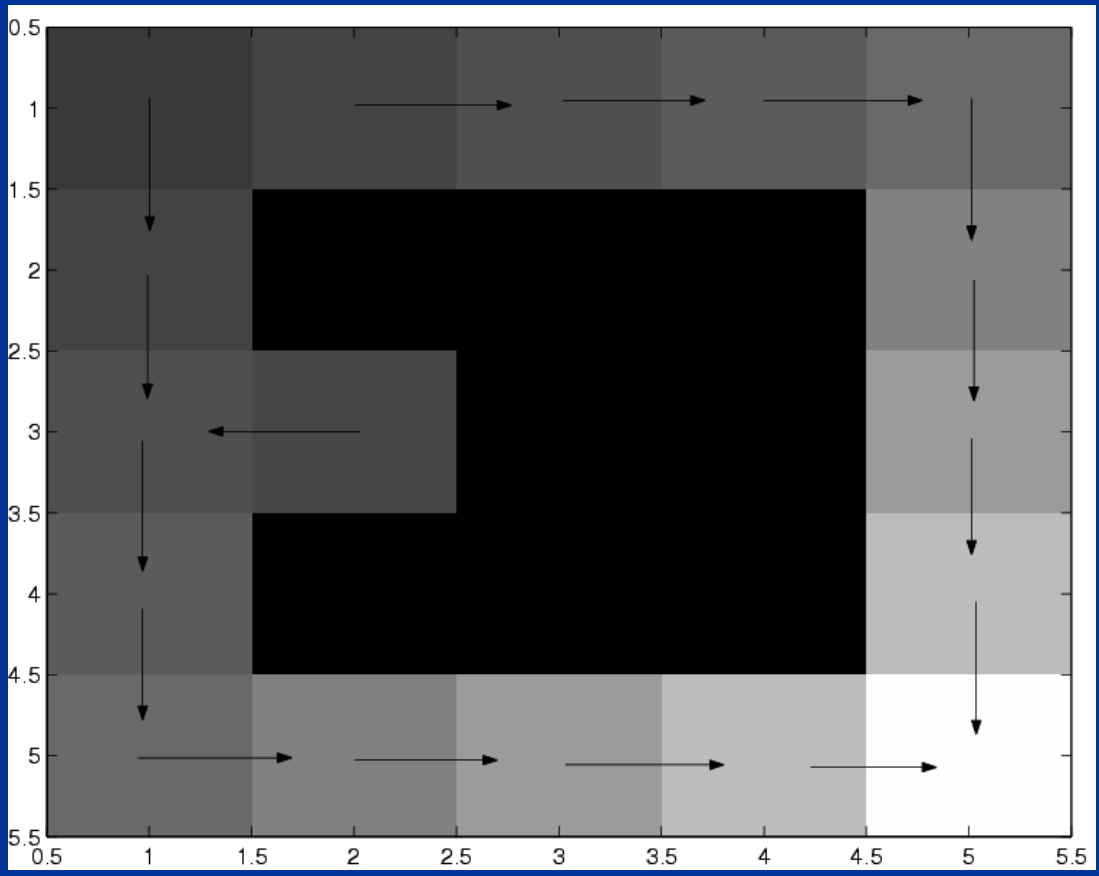
Ejemplo: robot virtual



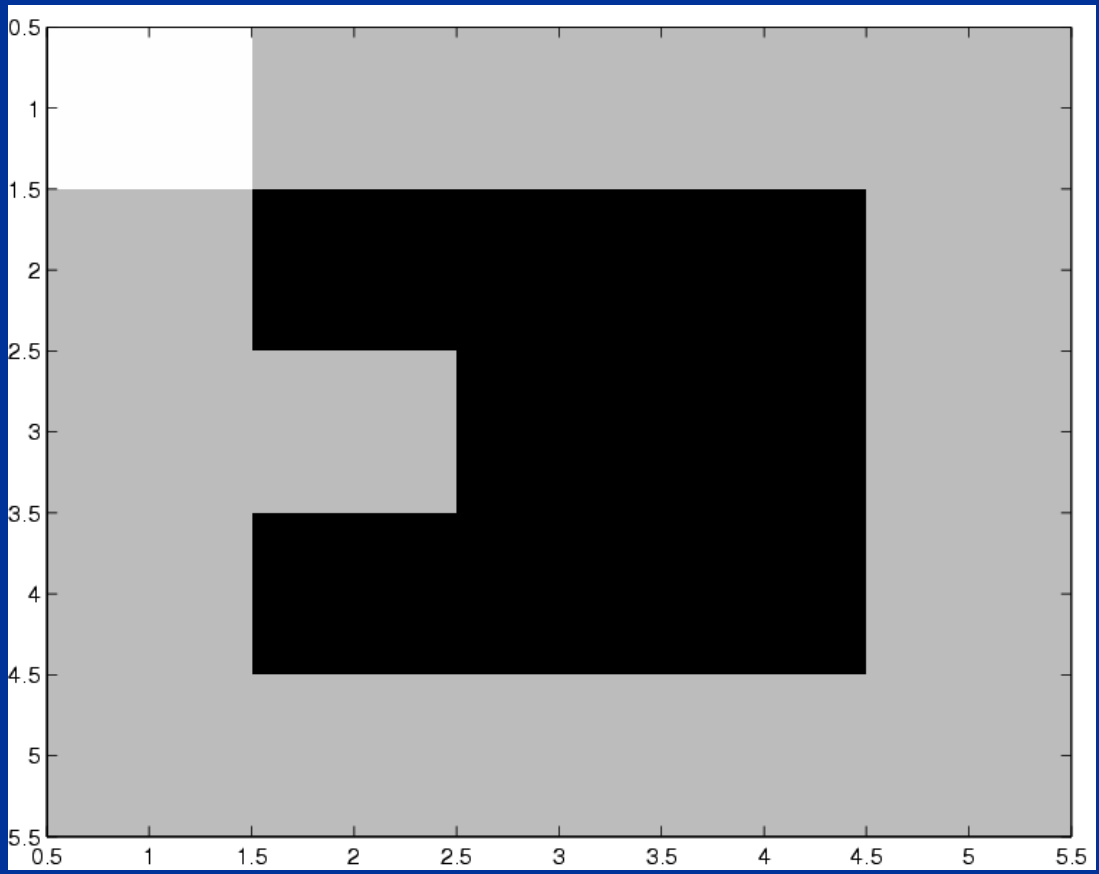
Una configuración



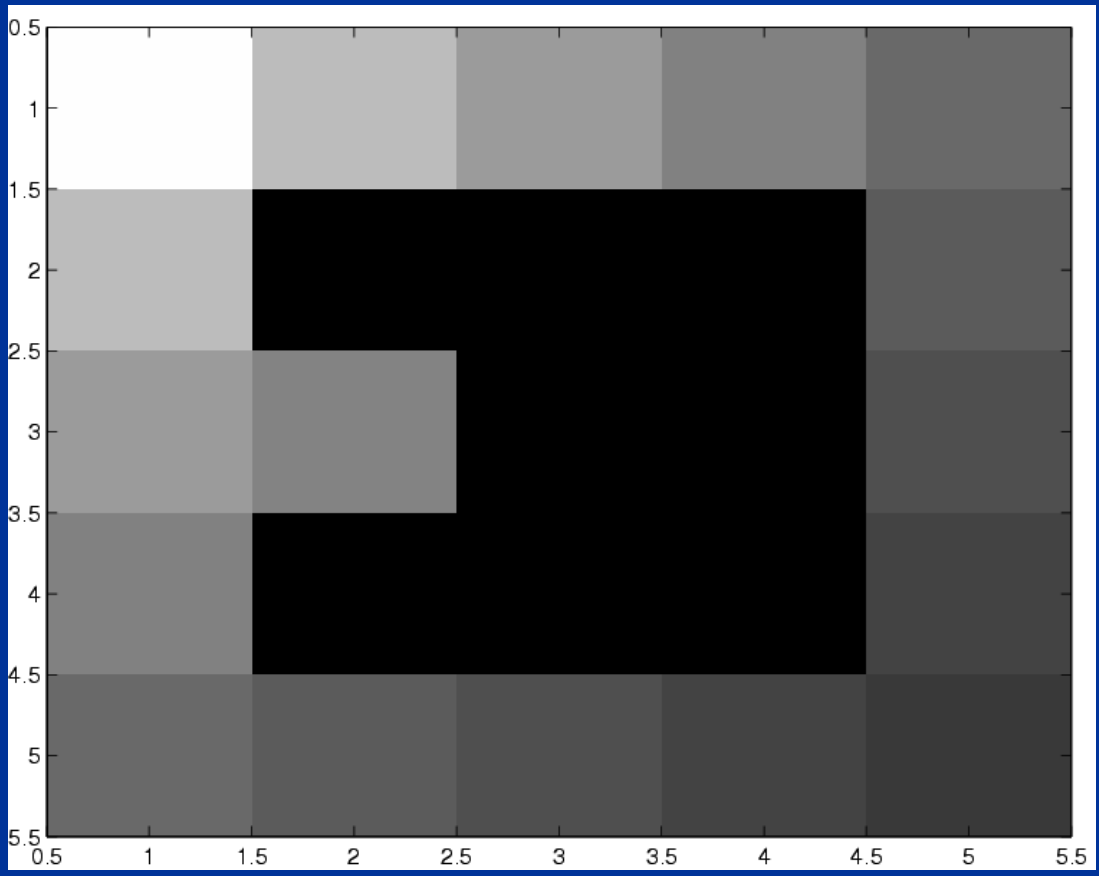
Política óptima



Otra configuración



Función de valor



POMDP

- En muchos problemas reales, no se puede observar exactamente el estado del agente, por lo que se tiene un POMDP
- Además de los elementos de un MDP, un POMDP incluye:
  - Una función de observación que especifica la probabilidad de las observaciones dado el estado, $P(O|S)$
  - Una distribución de probabilidad inicial para los estados, $P(S)$

POMDP

- El enfoque exacto para resolver un POMDP requiere considerar toda la historia de observaciones y acciones
- Esto es equivalente a considerar la distribución de probabilidad sobre los estados y en base a esta determinar las decisiones óptimas
- Para ello, se puede considerar un POMDP como un MDP en que los estados corresponden a la distribución de probabilidad
- El problema es que el espacio de estados se vuelve infinito y la solución exacta es muy compleja

POMDP

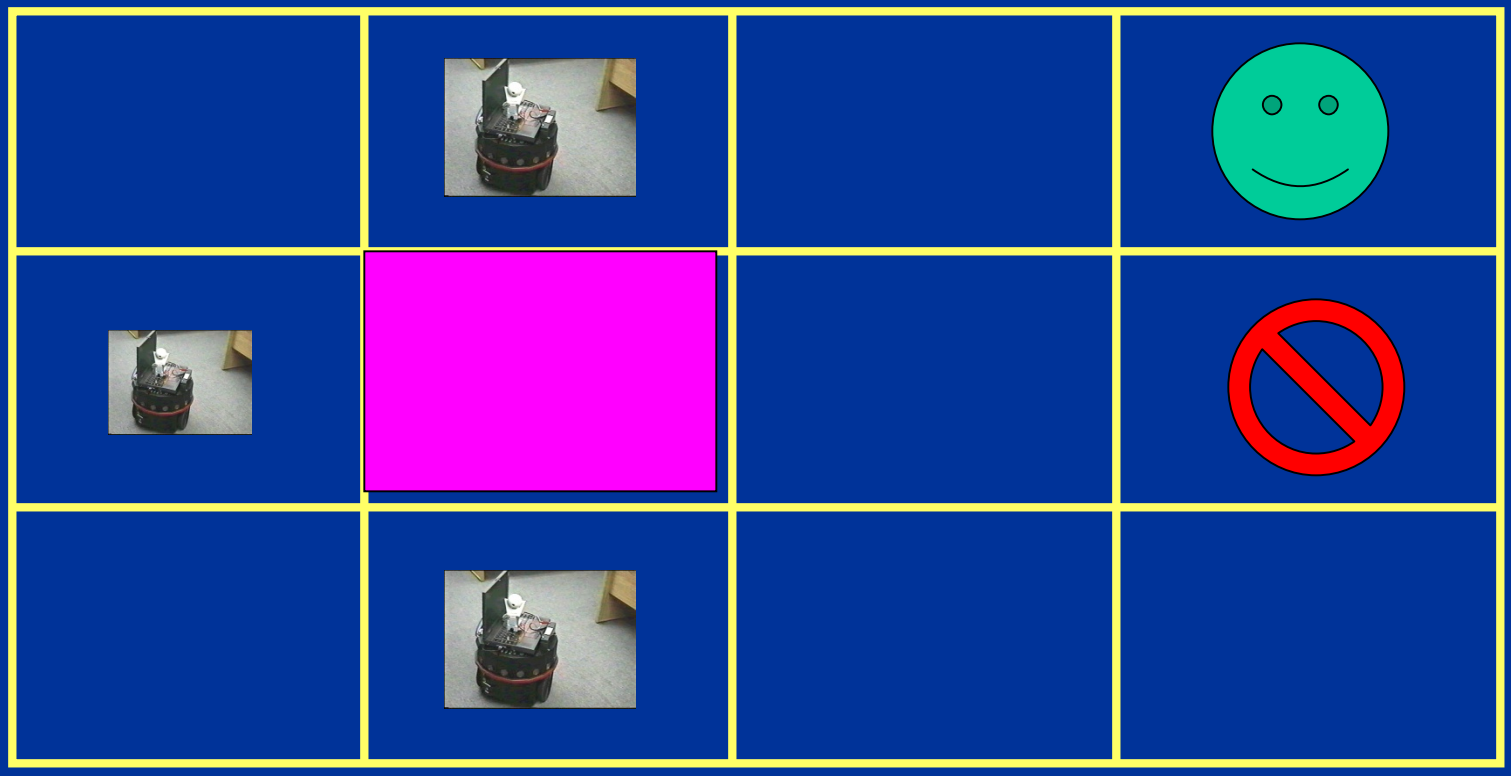
- Soluciones aproximadas:
  - Representar un POMDP de horizonte finito a través de un "policy tree" (acciones | observaciones | acciones …), resolviendolo en forma recursiva a través de planes condicionales
  - Para POMDP de horizonte finito la función de valor es *convexa y lineal a pedazos* (vectores α). Se puede aproximar mediante un subconjunto de vectores que dominan a los demás, y mediante esto encontrar una política aprox. óptima.

POMDP

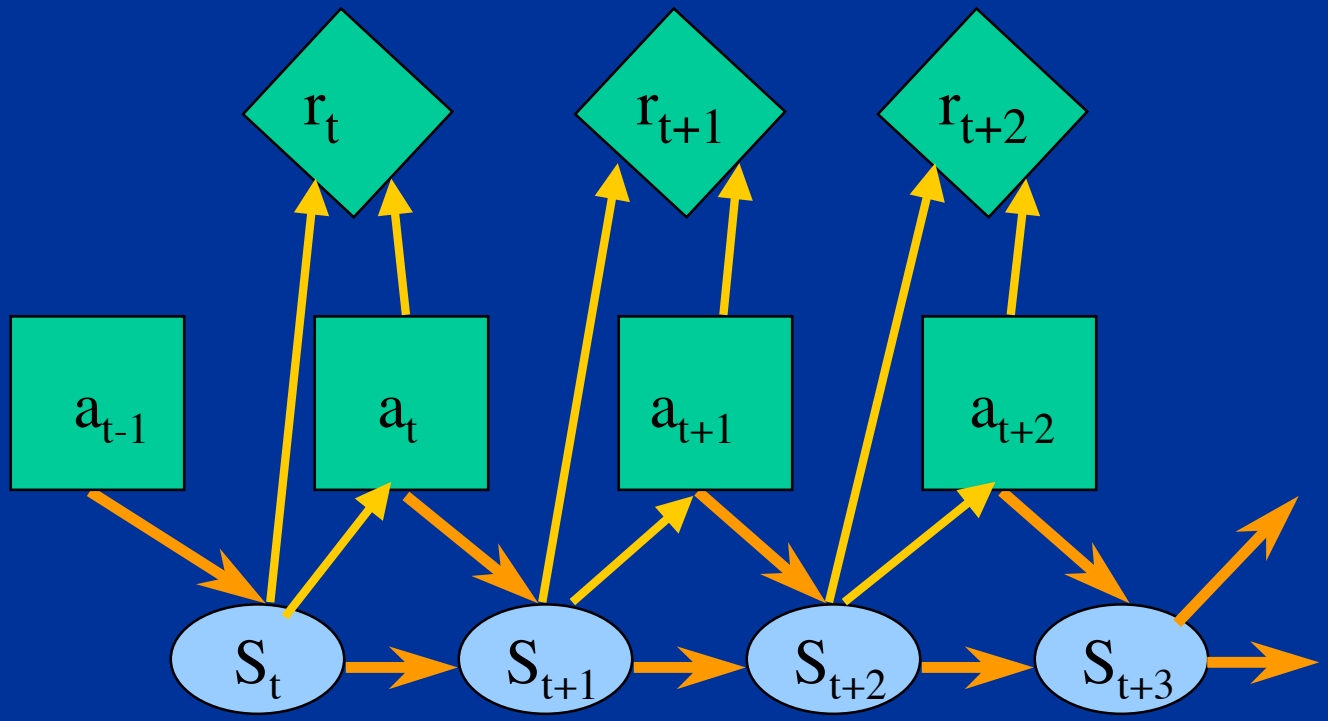
- Soluciones aproximadas:
  - Considerar un número finito de pasos y modelar el problema como una red de decisión dinámica – la aproximación depende del número de estados que se "ven" hacia delante (*lookahead*)

Ejemplo POMDP

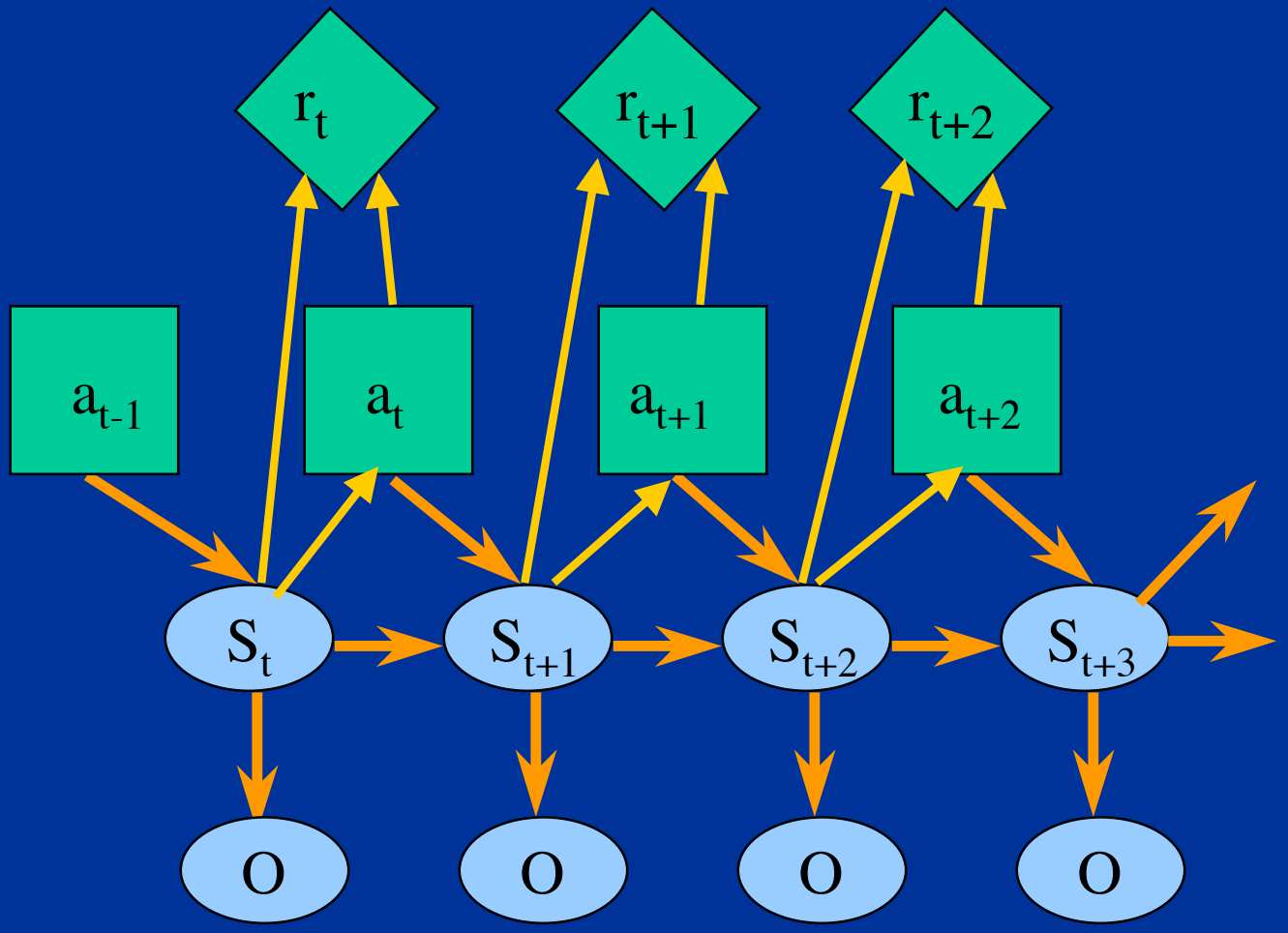
- El robot detecta su posición con sonares
  - Hay errores y ruido en las lecturas, alcance limitado
  - Ciertas celdas son muy parecidas (1,2 – 3,2)



MDP como una RDD



POMDP como una RDD



Extensiones

El principal problema de los MDPs es el crecimiento de la complejidad al aumentar el número de estados y acciones (el tamaño del problema crece exponencialmente con el número de variables de estado y acciones). Para ello se han planteado:

- Representaciones factorizadas
- Representaciones abstractas (agregación de estados)
- Modelos jerárquicos (serie / paralelo)

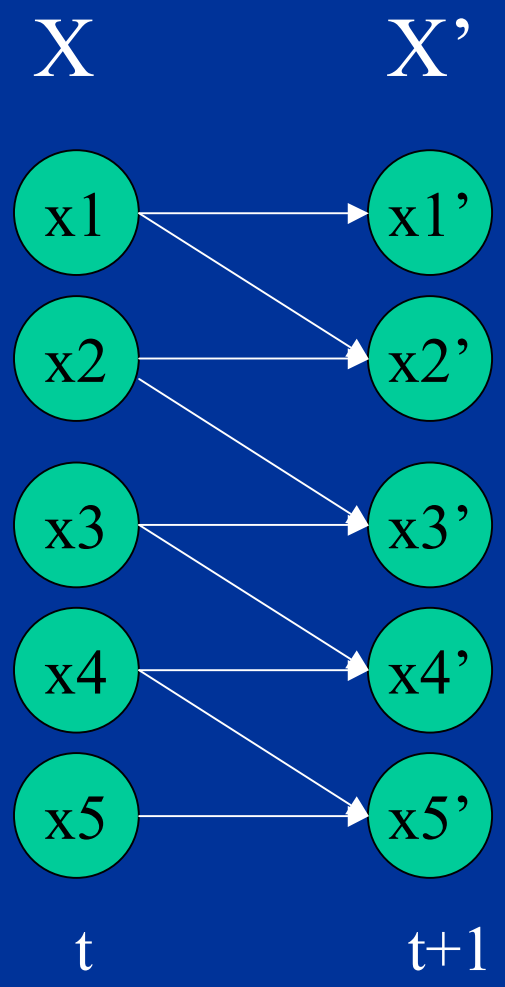
MDPs factorizados

- El estado de descompone en un conjunto de variables o factores
- Esto permite utilizar representaciones basadas en modelos gráficos para reducir la complejidad del modelo de transición y de recompensa:
  - El modelo de transición se representa usando RBD (una RBD de 2 etapas por acción)
  - La función de recompensa se representa usando árboles de decisión (considerando sólo las variables relevantes)

MDP factorizado

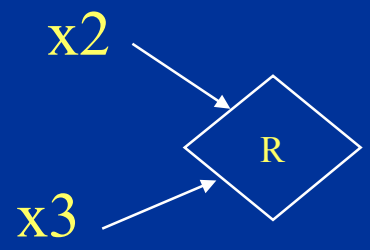
$X = \{x1, x2, x3, x4, x5\}$

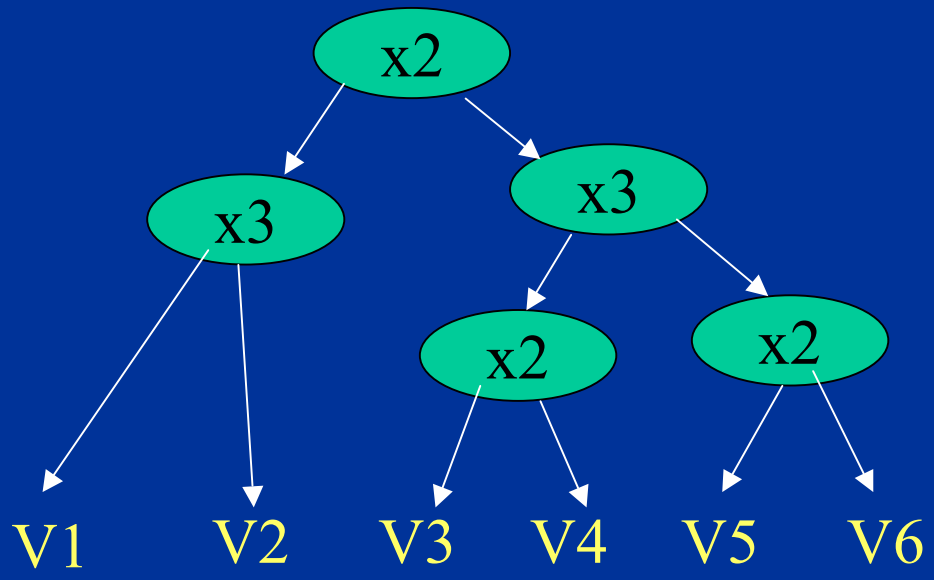
Se tiene una RBD por acción



MDP - factorizado

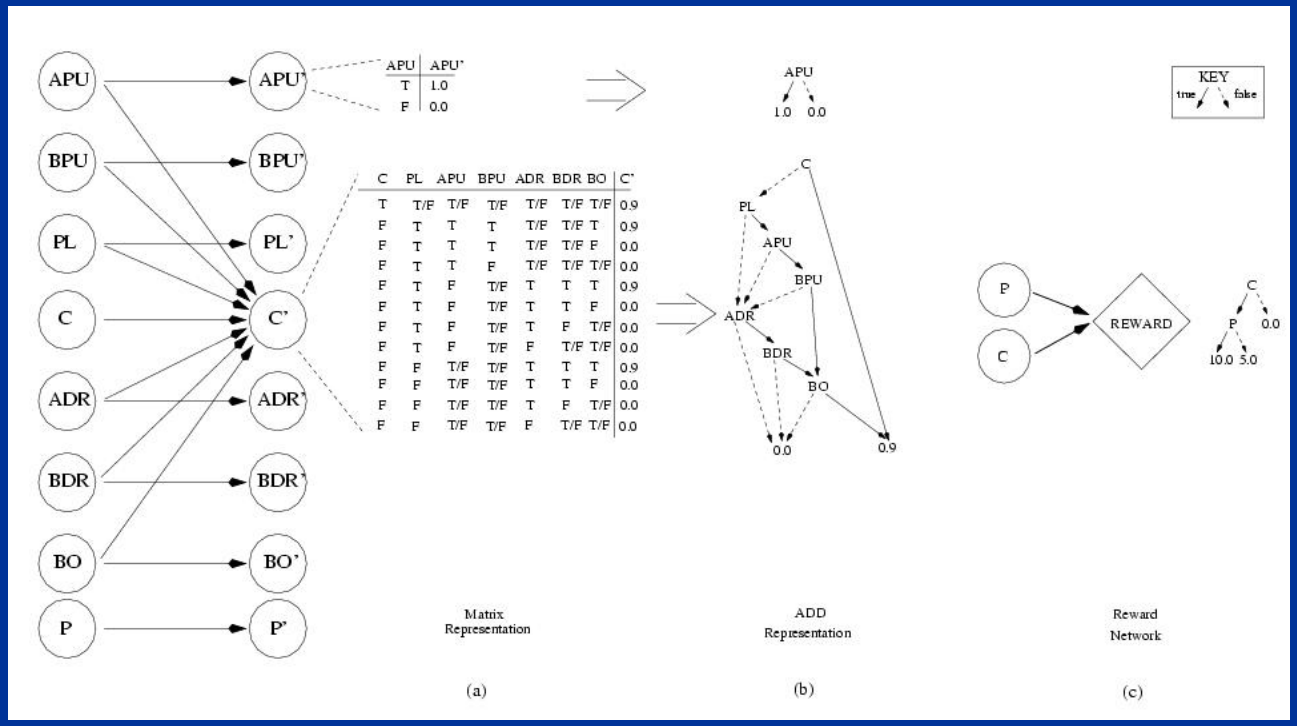
La función de recompensa considera sólo las variables que inciden directamente





Diagramas de Decisión Algebraicos

–Otra alternativa para representar en forma compacta M y R es mediante Diagramas de Decisión Algebraicos (SPUDD)



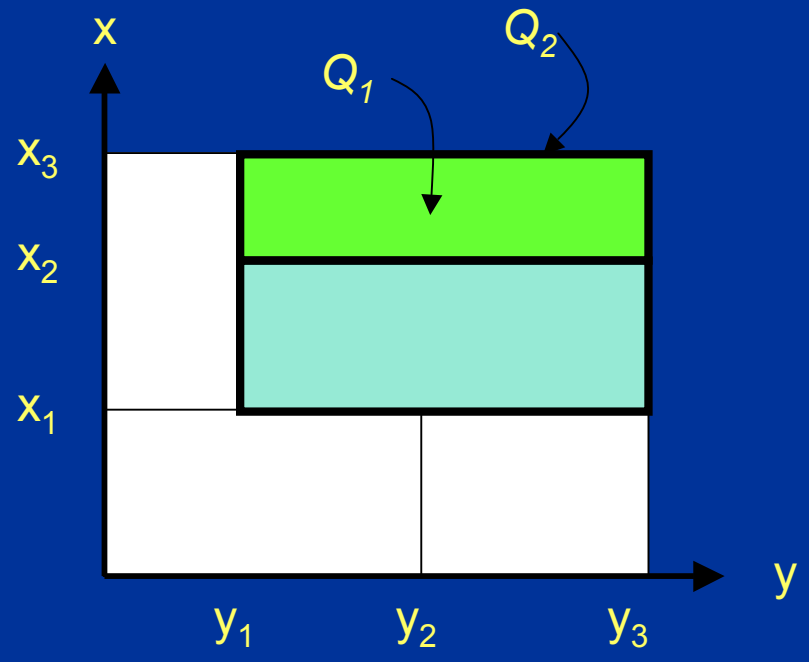
SPUDD

- SPUDD [Hoey et al., 1999] es un sistema que utiliza ADDs para resolver eficientemente MDPs muy grandes
- Utiliza algoritmos muy eficientes para hacer operaciones con ADDS (desarrollados en el área de diseño y verificación de circuitos electrónicos)

MDPs abstractos

- Otra alternativa para reducir la complejidad es reducir el número de estados, agrupando estados con propiedades similares (recompensa, probabilidades de transición)
- Esto también se puede aplicar a dominios continuos
- La desventaja es que la solución puede ya no ser óptima

Estados abstractos (cualitativos)



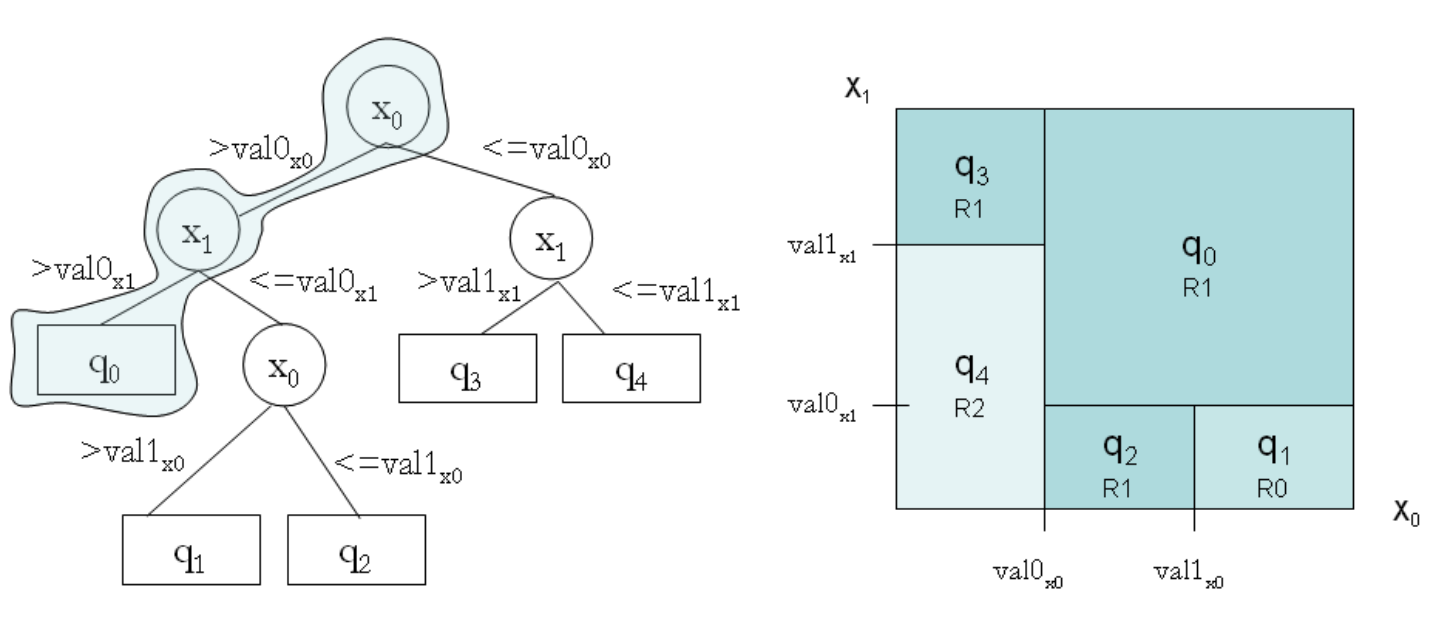
x_1, x_2, x_3, y_1, y_2, y_3 son valores de referencia o corte

Q_1=*pos(x, x2), ~pos(x,x3), pos(y, y1), ~pos(y,y3).*

Q_2=*pos(x, x1), ~pos(x,x2), pos(y, y1), ~pos(y,y3).*

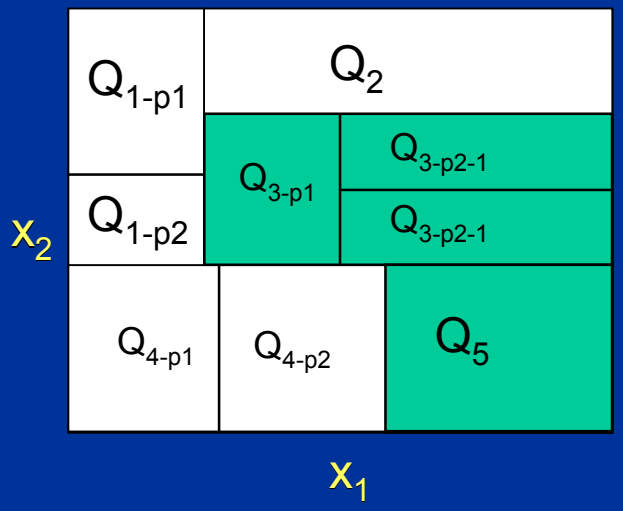
Partición cualitativa

Una partición cualitativa *q* es un grupo de estados con recompensas similares.



Refinamiento

- Si la política obtenida no es satisfactoria, se pueden agregar particiones adicionales o desagrupar estados.



Descomposición

- La otra alternativa para simplificar la solución de un MDP es "partir" el problema en subproblemas, de forma que se puede resolver c/u por separado y después "integrar la solución"
- Dos principales enfoques:
  - serie: se descompone la tarea en subtareas de forma que cada es una submeta que hay que cumplir para alcanzar la meta global (p. ej. *Heirarchical RL*)
  - paralelo: se descompone el problema en subproblemas que puedan resolverse "independientemente" y ejecutarse en "paralelo" (p. ej. *Parallel MDPs, Concurrent MDPs*)

Aprendizaje de MDPs

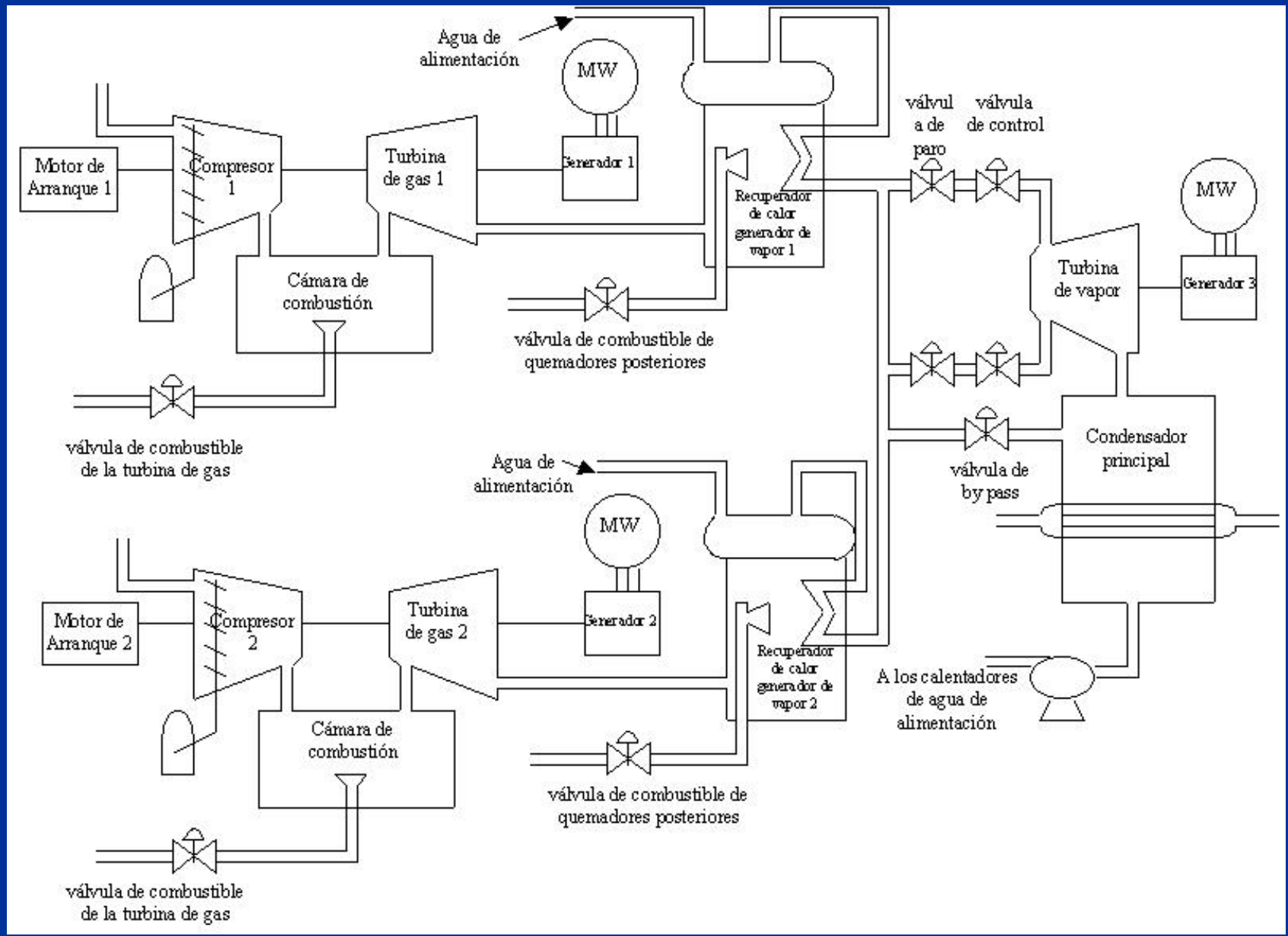
- Aprender el modelo:
  - En base a una exploración aleatoria del ambiente, se puede aprender el modelo de transición y la función de recompensa
- Sin modelo:
  - Se aprende directamente la política explorando el ambiente (aprendizaje por refuerzo: *Q-learning*)
- Enfoques híbridos:
  - *Dyna-Q, priotarized sweeping, …*

Aplicaciones

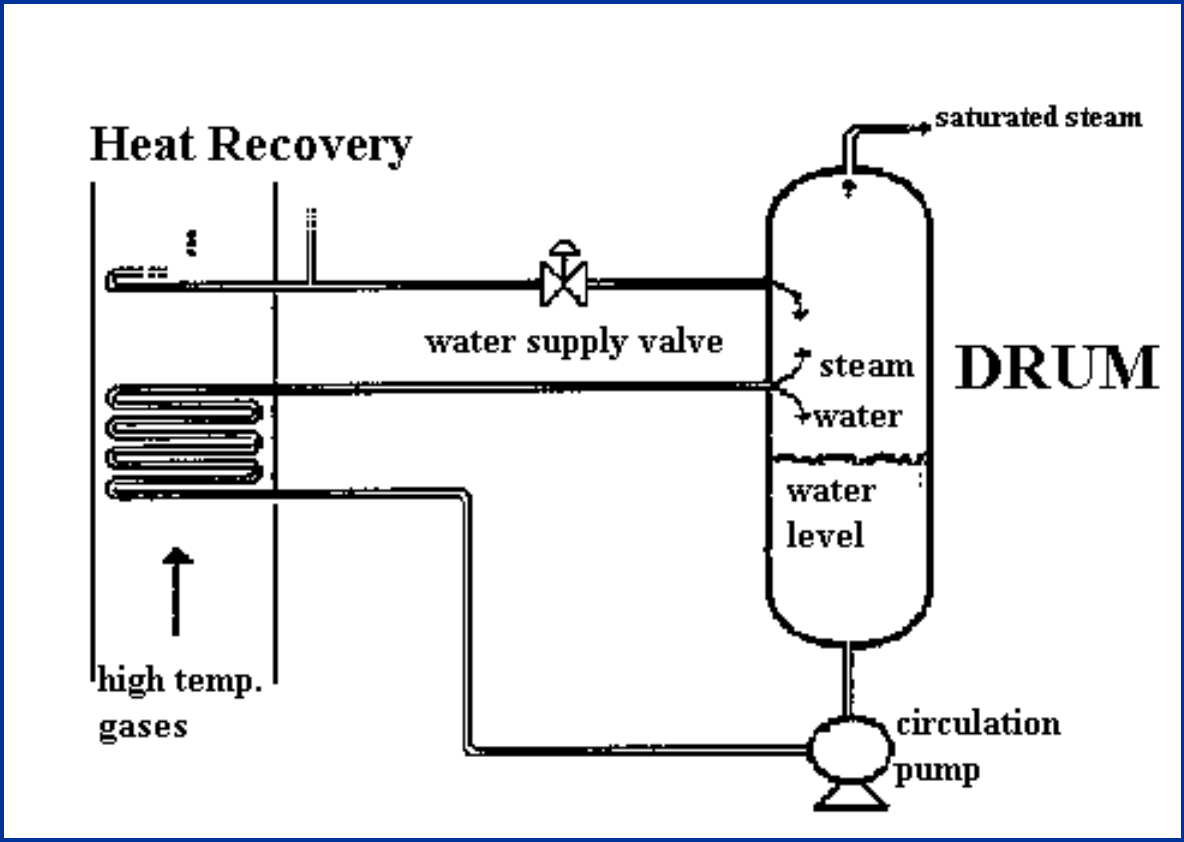
- Manejo de inventarios
- Mantenimiento de equipos y carreteras
- Control de sistemas de comunicaciones
- Modelado de procesos biológicos
- Planeación en robótica móvil
- Construcción de mapas / localización
- Control de procesos industriales
- Control de aviones
- …

Ejemplo de Aplicación

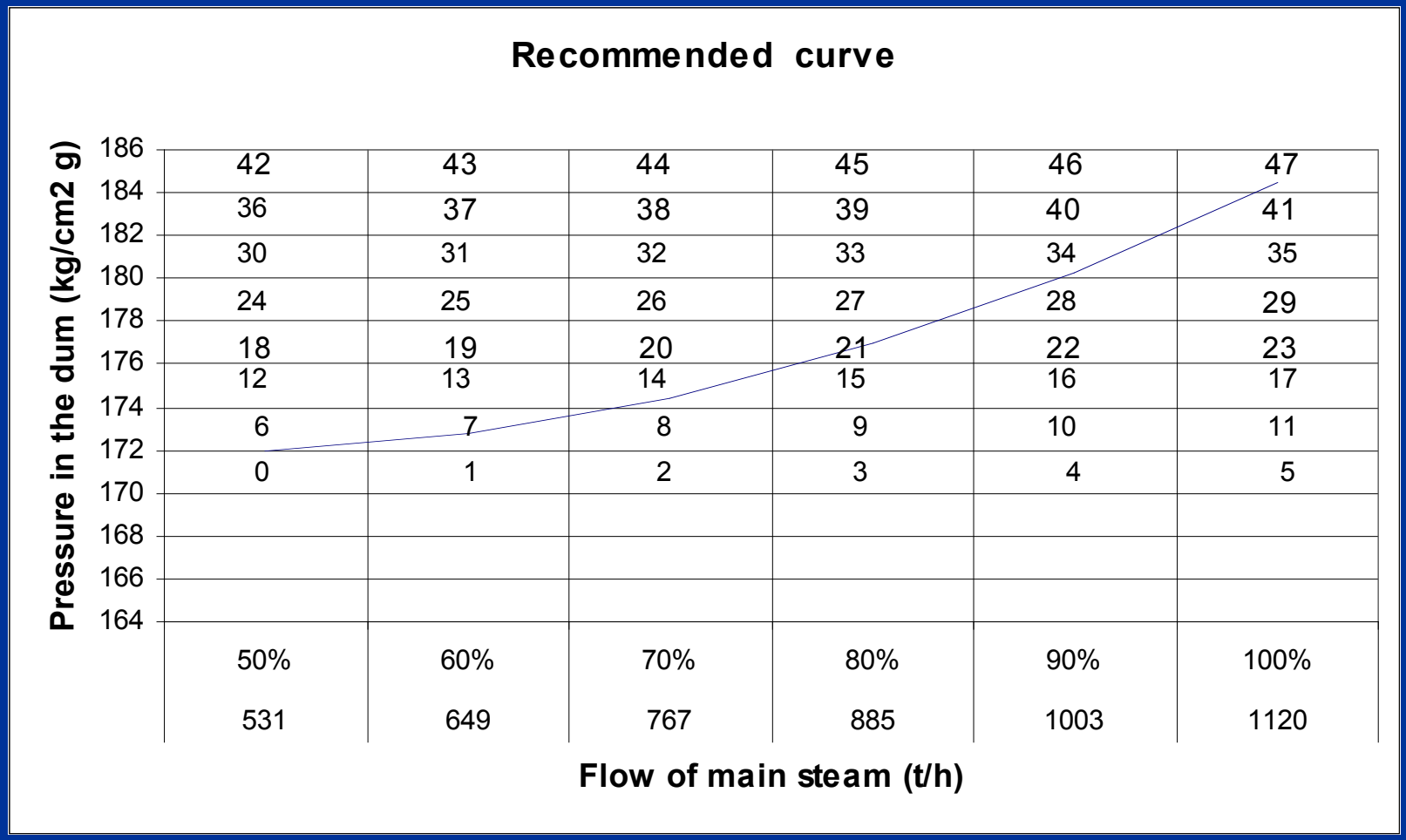
Control de una Planta Eléctrica utilizando MDP


Agua de alimentación
MW
válvul a de paro
válvula de control
Motor de Arranque 1
Compresor 1
Turbina de gas 1
Generador 1
Recuperador de calor generador de vapor 1
MW
Turbina de vapor
Generador 3
Cámara de combustión
válvula de combustible de quemadores posteriores
válvula de combustible de la turbina de gas
Agua de alimentación
válvula de by pass
Condensador principal
MW
Motor de Arranque 2
Compresor 2
Turbina de gas 2
Generador 2
Recuperador de calor generador de vapor 2
A los calentadores de agua de alimentación
Cámara de combustión
válvula de combustible de quemadores posteriores
válvula de combustible de la turbina de gas

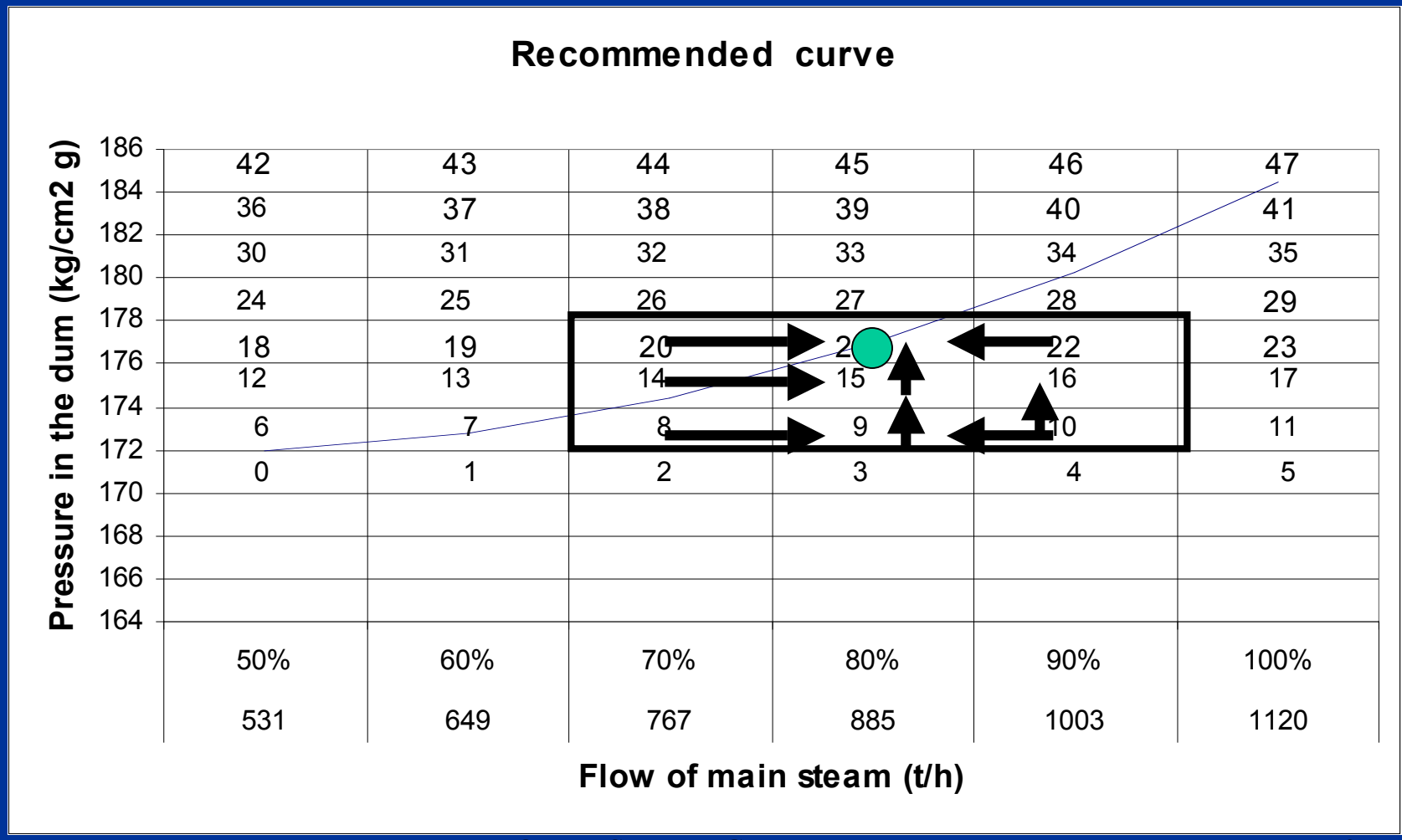
Generador de vapor y domo



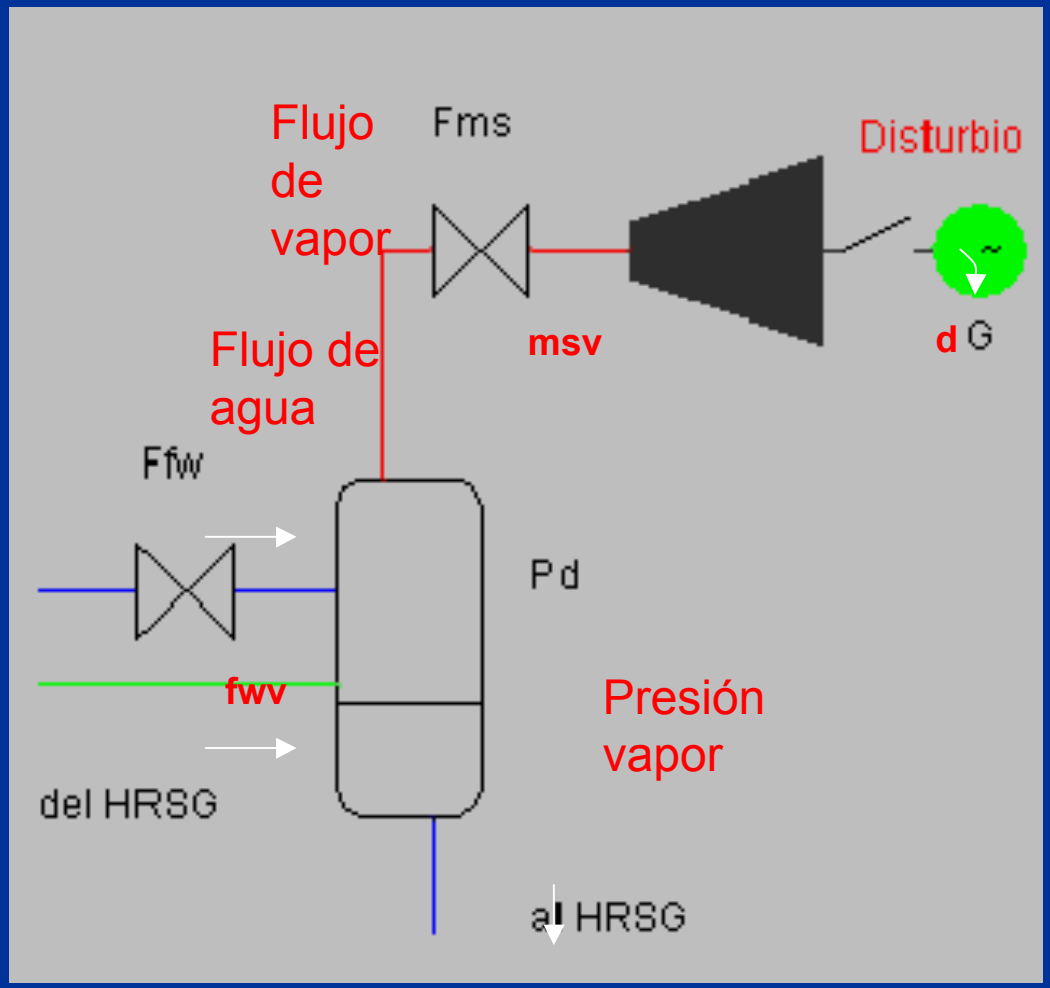
Espacio de control



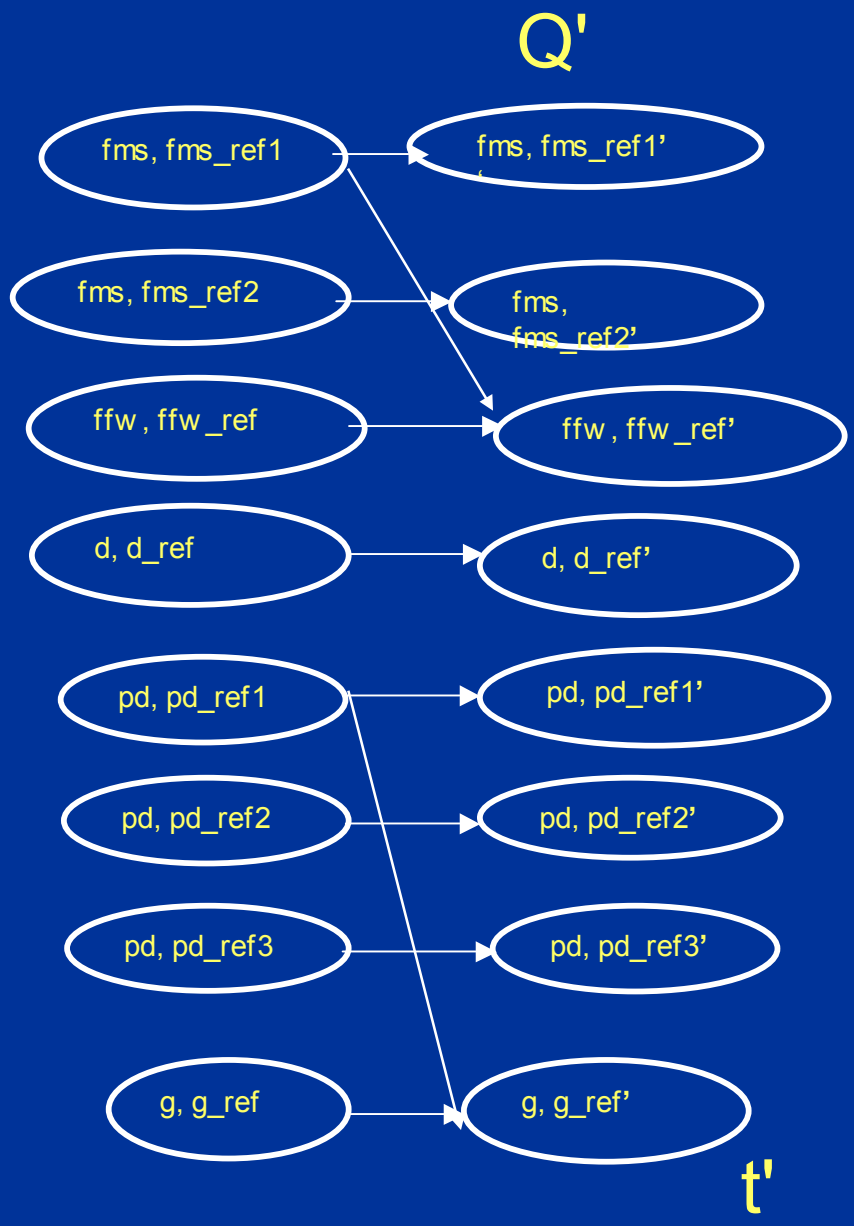
Ejemplo acciones



Variables relevantes



Modelo de Transición

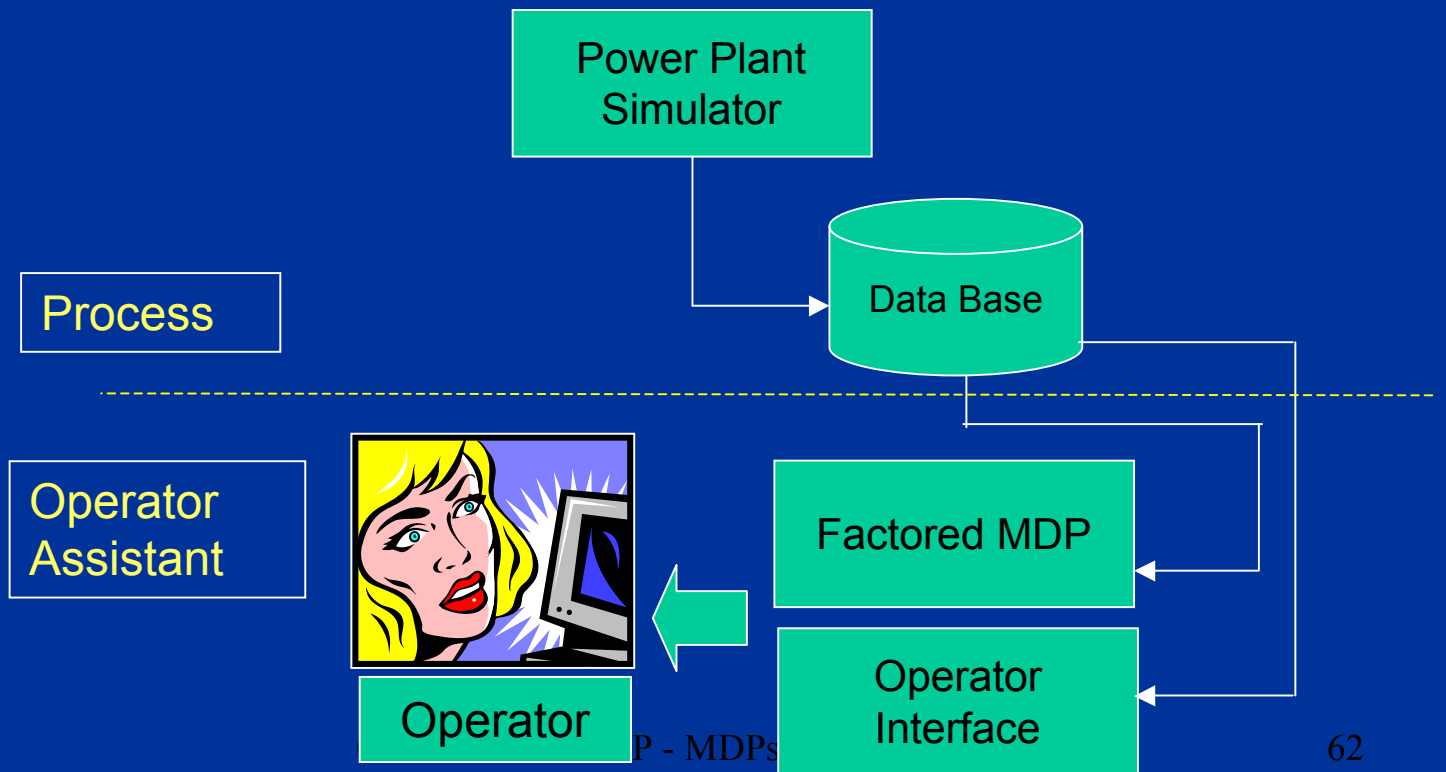


	0	+	-
0	0.33	0.13	0.01
+	0.33	0.82	0.00
-	0.33	0.05	0.99

acción: cerrar válvula

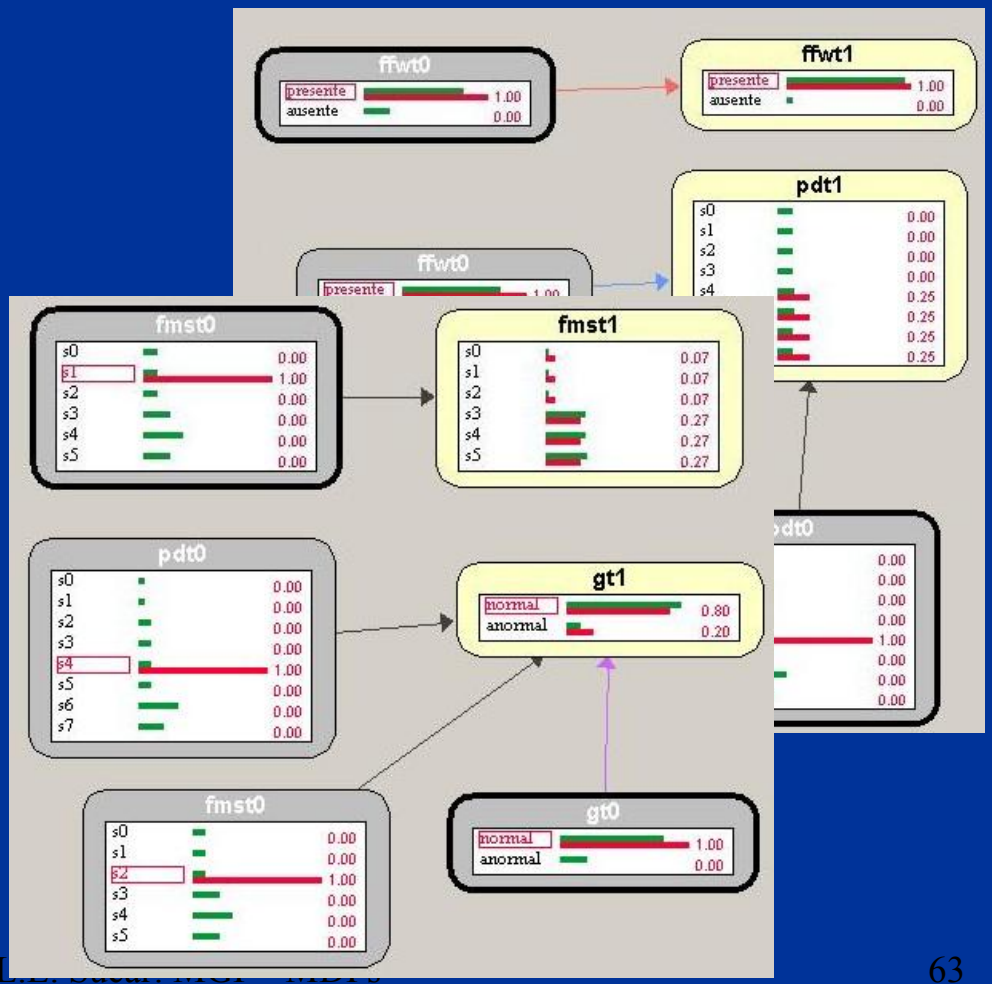
Asistente del Operador

- Arquitectura



Resultados

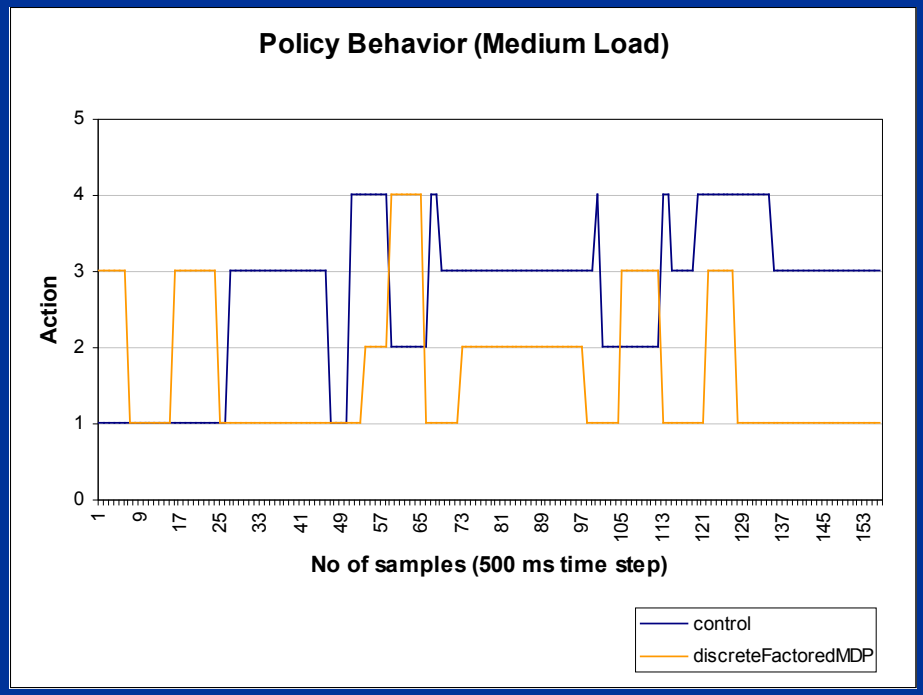
- El modelo de transición se obtuvo de a partir del simulador



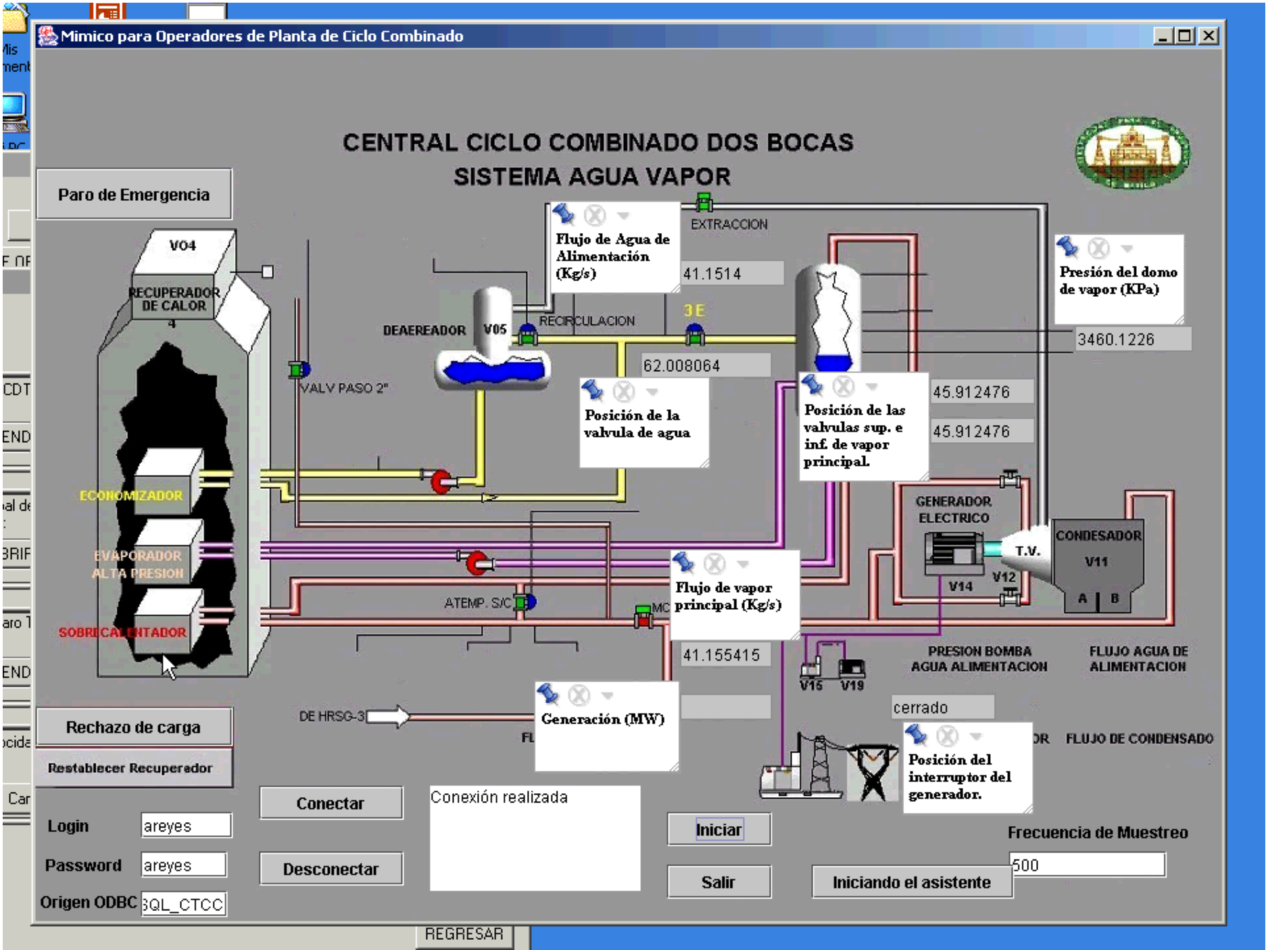
Resultados – comparación de un MDP plano con uno factorizado

Parámetros	a0	a1	a2	a3	Total	Tiempo de solución
MDP "plano"	147456	147456	147456	147456	589824	5.6 días
MDP factorizado	175	175	204	204	758	2 minutos

Resultados – comparación con el control convencional



En algunos casos son similares, pero el MDP lleva más rápidamente a la planta a un estado seguro.


Mimico para Operadores de Planta de Ciclo Combinado
CENTRAL CICLO COMBINADO DOS BOCAS
SISTEMA AGUA VAPOR
Paro de Emergencia
V04
RECUPERADOR DE CALOR
4
ECONOMIZADOR
EVAPORADOR ALTA PRESION
SOBRECALENTADOR
DEAEREADOR
V05
RECIRCULACION
VALV PASO 2"
ATEMP. S/C
MC
DE HRSG-3
EXTRACCION
Flujo de Agua de Alimentación (Kg/s)
41.1514
3E
62.008064
Posición de la valvula de agua
Posición de las valvulas sup. e inf. de vapor principal.
45.912476
45.912476
Presión del domo de vapor (KPa)
3460.1226
GENERADOR ELECTRICO
T.V.
CONDESADOR
V11
A
B
V14
V12
Flujo de vapor principal (Kg/s)
41.155415
PRESION BOMBA AGUA ALIMENTACION
FLUJO AGUA DE ALIMENTACION
V15
V19
cerrado
Generación (MW)
FLUJO DE CONDENSADO
Posición del interruptor del generador.
Rechazo de carga
Restablecer Recuperador
Conectar
Conexión realizada
Login
areyes
Password
areyes
Origen ODBC
SQL_CTCC
Desconectar
Iniciar
Salir
Iniciando el asistente
Frecuencia de Muestreo
500
REGRESAR

Ejemplo de Aplicaciones

Coordinación de tareas para un robot de servicio - Markovito

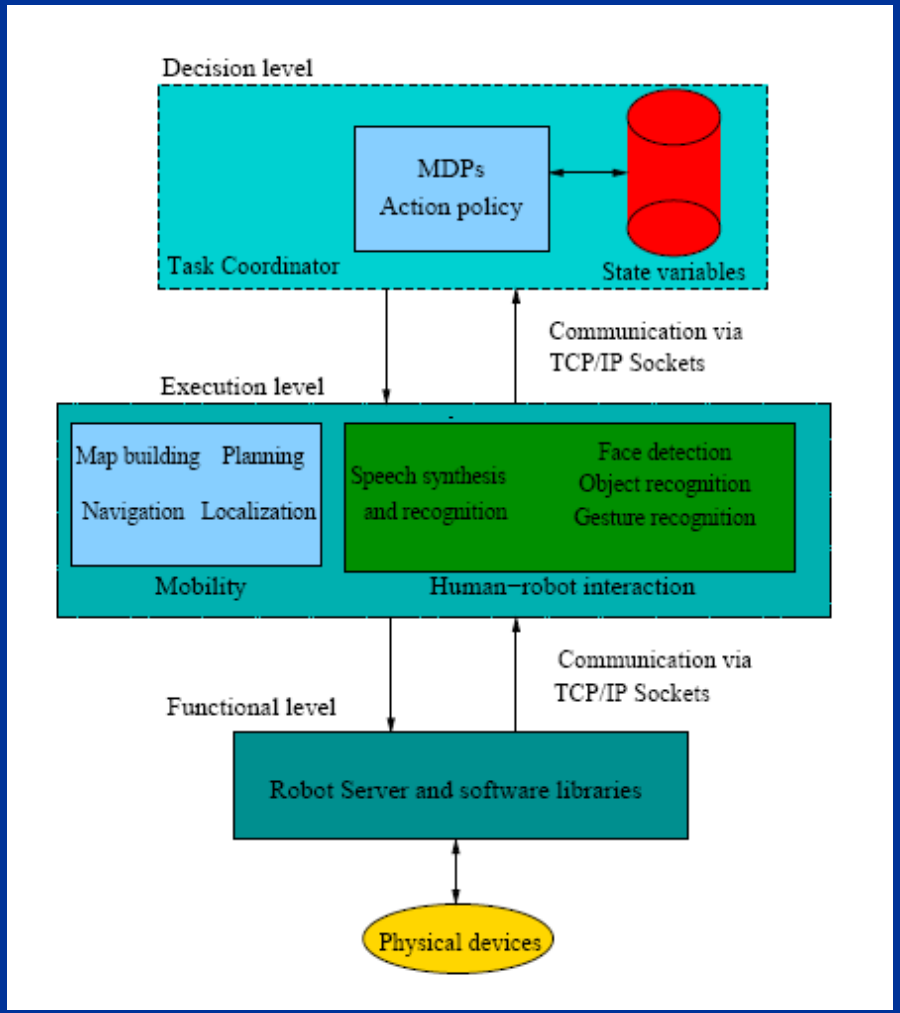
Coordinación de Tareas para Robots de Servicio

- Una tarea compleja en robótica de servicio requiere de diversas habilidades:
  - Planeación de trayectorias
  - Evitar obstáculos
  - Localización
  - Construcción de mapas
  - Encontrar personas
  - Reconocimiento y síntesis de voz
  - Generación de expresiones
  - …

Coordinación

- La coordinación de los diferentes módulos para realizar una tarea se base en un MDP
- De acuerdo a la tarea se define una función de recompensa, y al resolver el MDP se obtiene la política óptima para dicha tarea
- De esta forma diversos módulos se pueden re-utilizar para diferentes tareas sólo cambiando el MDP

Arquitectura de Software



Markovito: hardware

- Robot PeopleBot
- Cámara Pan/tilt
- Micrófono direccional
- 2 anillos de sonares
- Pinza
- Laser
- 2 computadoras (interna & laptop)
- Monitor con “cara animada”

Aplicaciones

- Basado en este enfoque se han desarrollado diversas tareas para un robot de servicio:
  - **Robot mensajero**: lleva mensaje u objetos de una persona a otra
  - **Robot anfitrión**: recibe visitantes y los guía en una institución
  - **Navegación** (Robocup@home): va a diferentes lugares en una casa comandado por voz
  - **Busca y encuentra** (Robocup@home): busca un objeto
  - **Seguimiento** (Robocup@home): sigue a una persona

Navegación en un ambiente de “casa”

Mensajero

Enseñando a Markovito a Reconocer un objeto

Entregando una cerveza!

MDPs Concurrentes

- Se requiere que el robot pueda hacer diversas acciones al mismo tiempo, por ejemplo navegar, escuchar a una persona y sonreír (cara animada)
- Si se consideran todas las combinaciones de acciones en un MDP se tiene una explosión aún mayor en el número de estados-acciones
- Una alternativa es dividir el problema en varias sub-tareas, de forma que cada sub-tarea se representa con un MDP; y las políticas de cada sub-MDP se ejecutan en forma concurrente

MDPs Concurrentes

- Al descomponer el problema pueden existir conflictos entre las sub-tareas, que pueden ser de dos tipos:
  - Conflictos por recursos
  - Conflictos por comportamiento
- Los conflictos por recursos se resuelven en un proceso de dos fases, en la primera se resuelve cada subMDP en forma independiente y se construye una política inicial combinada; luego la política se afina usando iteración de políticas
- Los conflictos por comportamiento se resuelven a través de una serie de restricciones definidas por el usuario, y en línea se selecciona el conjunto de acciones de mayor valor que satisfacen las restricciones

Referencias

- [Russell & Norvig] – Cap. 17
- [Sucar, Morales, Hoey] - Cap. 3
- H. A. Taha, “Investigación de Operaciones”, Alfaomega, 1991 – Cap. 14
- M. Puterman, “Markov Decision Processes”, Wiley, 1994.

Referencias

- Blythe, J., 1999, Decision –Theoretic Planning. AAAI. AI Magazine, 37-54.
- C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: structural assumptions and computational leverage. Journal of Artificial Intelligence Research, 11:1–94, 1999
- D. Suc and I. Bratko. Qualitative reverse engineering. In Proceedings of the 19th International Conference on Machine Learning, 2000.
- E. Morales. Scaling up reinforcement learning with a relation representation.pages 15–26. Proc. of the Workshop on Adaptability in Multi-agent Systems (AORC-2003), 2003.

Referencias

- J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. Spudd: Stochastic planning using decision diagrams. In Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, UAI-99, pages 279–288, 1999.
- K. Forbus. Qualitative process theory. Artificial Intelligence, 24, 1984.
- R.S. Sutton and A.G. Barto. Reinforcement Learning: An Introduction. 1998.

Referencias

- E. Corona, E. Morales, L.E. Sucar, Executing concurrent actions with concurrent Markov decision processes, ADPRL, IEEE, 2009.
- P. Elinas, E. Sucar, A. Reyes and J. Hoey; A decision theoretic approach to task coordination in social robots, IEEE International Workshop on Robots and Human Interactive Communications RO-MAN 04; Japan 2004. Demo Videos.
- A. Reyes, P. H. Ibarguengoytia, L. E. Sucar; Power Plant Operator Assistant: An Industrial Application of Factored MDPs; Mexican International Conference on Artificial Intelligence (MICAI-04); Mexico City; April 2004.

Referencias

- A. Reyes, L. E. Sucar, P. Ibarguengoytia; Power Plant Operator Assistant; Bayesian Modeling Applications Workshop in the 19th Conference on Uncertainty in Artificial Intelligence UAI-03, Acapulco-Mexico, August 2003.
- A. Reyes, M.A. Delgadillo, P. H. Ibarguengoytia; An Intelligent Assistant for Obtaining the Optimal Policy during Operation Transients in a HRSG; 13th Annual Joint ISA POWID/ EPRI Controls and Instrumentation Conference; Williamsburg, Virginia, June 2003.
- Ibargüengoytia P. H., Reyes A. 2001. Continuous Planning for The Operation of Power Plants, Memorias del Encuentro Nacional de Computación ENC 2001, Aguscalientes-Mexico.

Software

- MDPs
  - **Markov Decision Process (MDP) Toolbox v1.0 for MATLAB** (INRIA) http://www.inra.fr/bia/T/MDPtoolbox/
  - **Markov Decision Process (MDP) Toolbox for Matlab (K. Murphy)** http://www.ai.mit.edu/~murphyk/Software/MDP/mdp.html
  - **SPUDD** http://www.cs.ubc.ca/spider/staubin/Spudd/