

Tema 1: Sistemas Inteligentes y Búsqueda

- Sistemas Inteligentes
- Búsqueda heurística y propiedades

Objetivos

- Entender las componentes básicas de un **Sistema Inteligente** a través de la descripción de la resolución de un problema mediante búsqueda y uso eficiente del conocimiento.
- Definir el concepto de **agente inteligente** y su aportación a la construcción de los Sistemas Inteligentes.
- Mostrar los distintos tipos de agentes y sus **arquitecturas**.
- Analizar el uso de la **heurística** en los **sistemas de búsqueda** y explicar las principales extensiones de los modelos básicos, detallando las ventajas e inconvenientes de cada extensión, junto con el contexto en donde es conveniente su aplicación.
- Estudiar las **propiedades formales** de los métodos heurísticos a través de los conceptos de admisibilidad y monotonía de las funciones heurísticas.

Estudia el tema en ...

- Nils J. Nilsson, *“Inteligencia Artificial: Una nueva síntesis”*, Ed. Mc Graw Hill, 2000.
- S. Russell, P. Norvig, *Artificial Intelligence: A modern Approach*, Tercera Edición, Ed. Pearson, 2010.

Sistemas Inteligentes

- Un **Sistema Inteligente** (SI) es un sistema software que muestra un comportamiento inteligente o interactúa de una forma más inteligente con su entorno que otros sistemas.
- La barrera entre un sistema software normal y un sistema inteligente queda un tanto difusa, al igual que la barrera entre los Sistemas Inteligentes y la Inteligencia Artificial.
- La **Inteligencia Artificial** se ocupa de la investigación básica en la implementación de cada una de las habilidades relacionadas con la inteligencia humana.
- Los **Sistemas Inteligentes** se encargan de aplicar la investigación, tratando de solucionar problemas ya existentes o mejorar nuestra forma de trabajar o calidad de vida aplicando técnicas de Inteligencia Artificial.

Problemas en Inteligencia Artificial

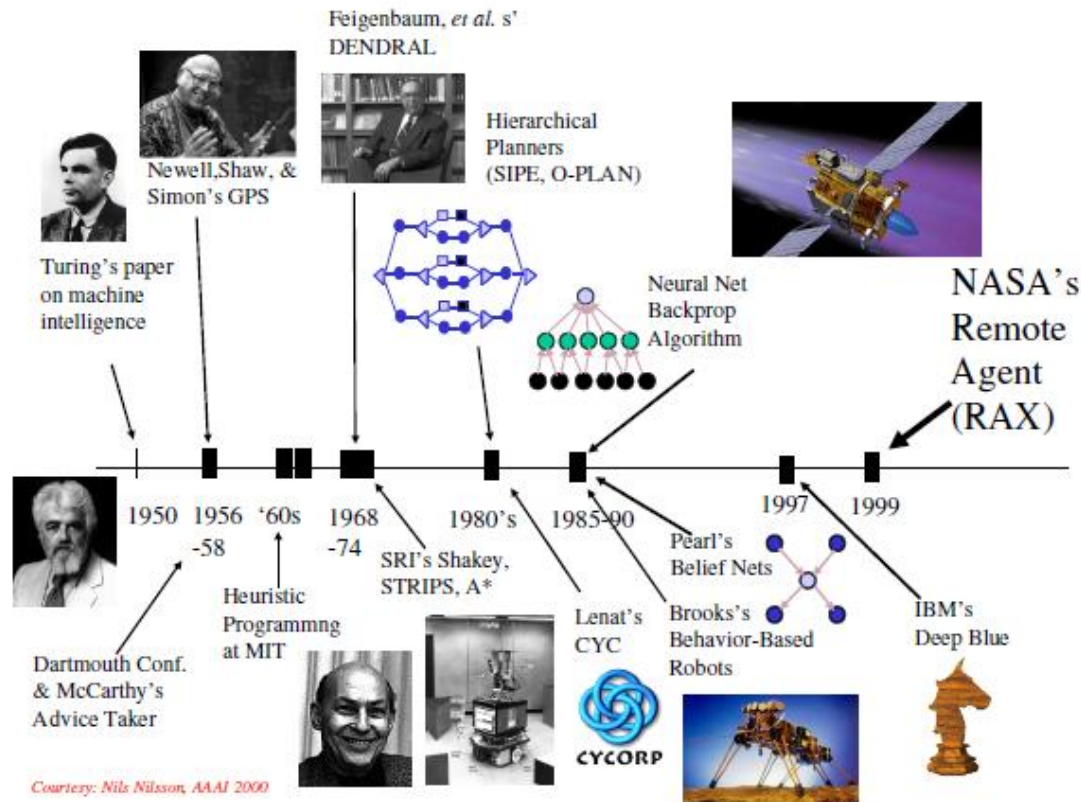
- Normalmente somos más *eficientes* que las máquinas en la resolución de problemas NP-duros

(Héctor Geffner, 2007)

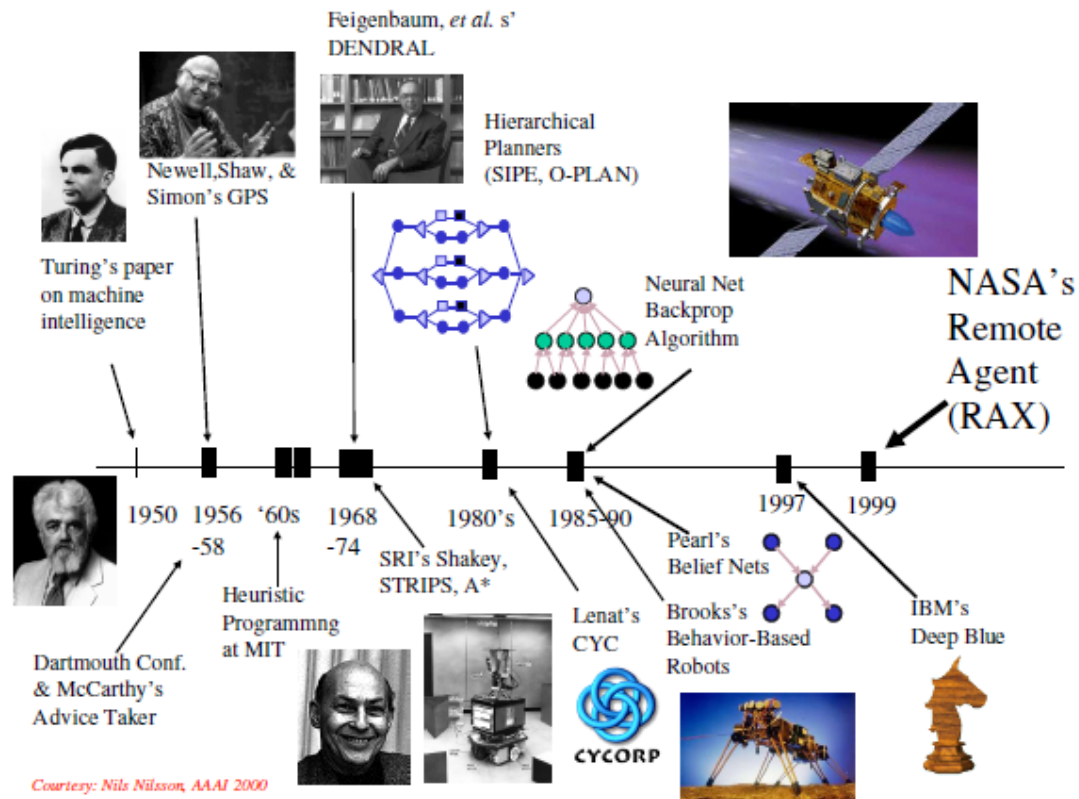
“Si un problema en Inteligencia Artificial tiene solución, ¡es que habíamos escogido mal el problema!”

- Pero hemos conseguido un progreso notable, ...

Progreso en el desarrollo de Sistemas Inteligentes



Progreso en el desarrollo de Sistemas Inteligentes



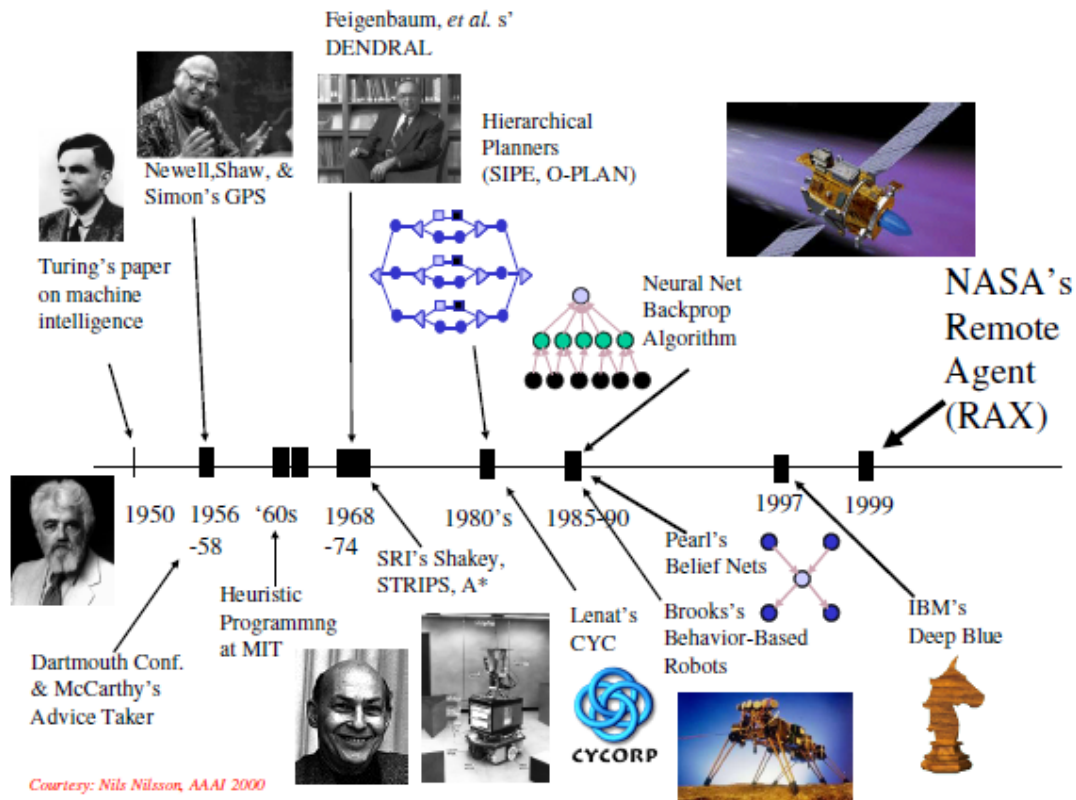
Alan Turing

1936: *The Turing machine, computability, universal machine*
1950: *The Turing Test for machine intelligence*

Claude Shannon

1950: Claude Shannon published detailed analysis of chess playing as search

Progreso en el desarrollo de Sistemas Inteligentes



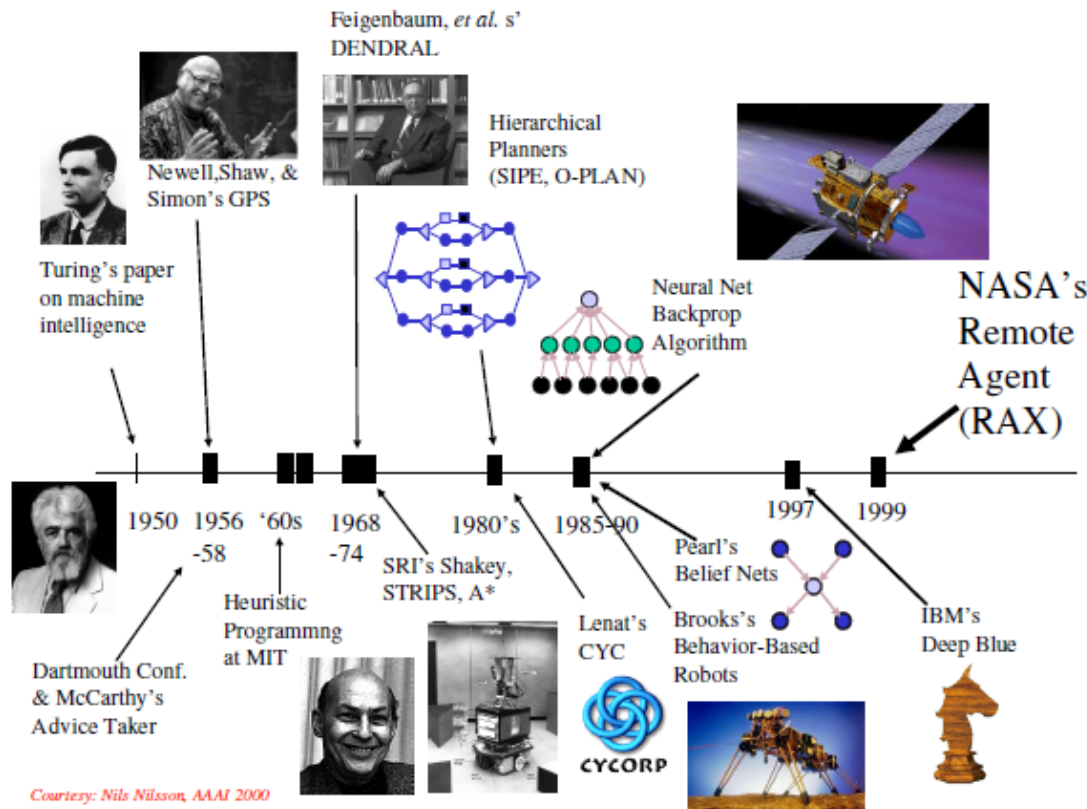
Darmouth Conference

1956: John McCarthy coined the term **Artificial Intelligence** as the topic of the Dartmouth Conference, the first conference devoted to the subject

Arthur Samuel (IBM)

1952–1962: wrote the first game-playing program, for checkers (draughts), to achieve sufficient skill to challenge a world champion. His first checkers-playing program was written in 1952, and in 1955 he created a version that learned to play (Samuel 1959)

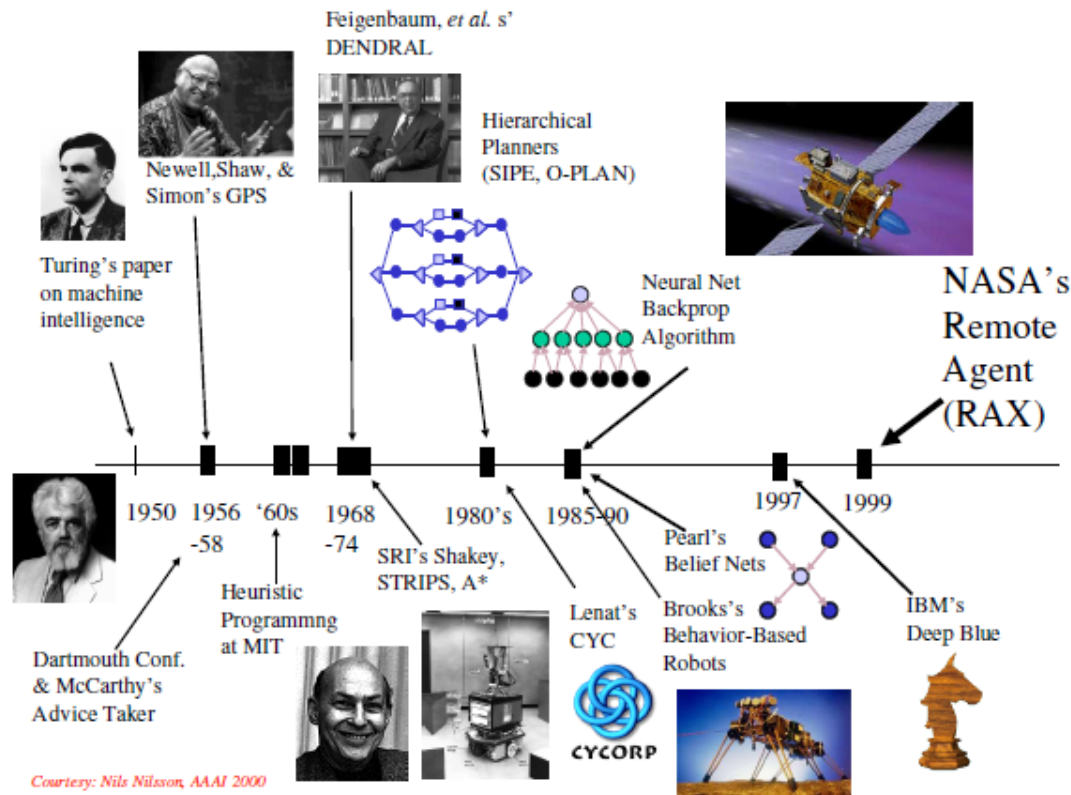
Progreso en el desarrollo de Sistemas Inteligentes



General Problem Solver

1957: **GPS** was a computer program created by Herbert Simon and Allen Newell to prove theorems and solve geometric, word and chess problems
1959: Newell, A.; Shaw, J.C.; Simon, H.A. *Report on a general problem-solving program*. Proceedings of the International Conference on Information Processing. pp. 256–264.

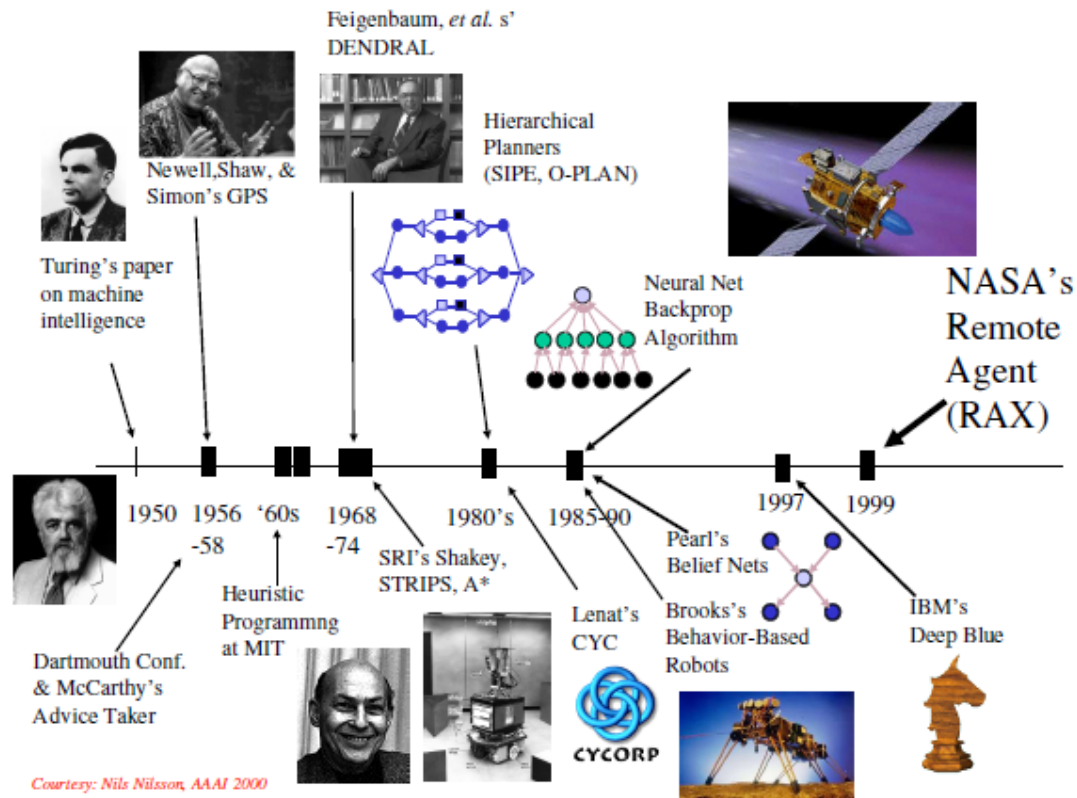
Progreso en el desarrollo de Sistemas Inteligentes



Heuristic Programming

1958: Teddington Conference on the Mechanization of Thought Processes was held in the UK and among the papers presented were John McCarthy's "*Programs with Common Sense*" Oliver Selfridge's "*Pandemonium*" and Marvin Minsky's "*Some Methods of Heuristic Programming and Artificial Intelligence*"

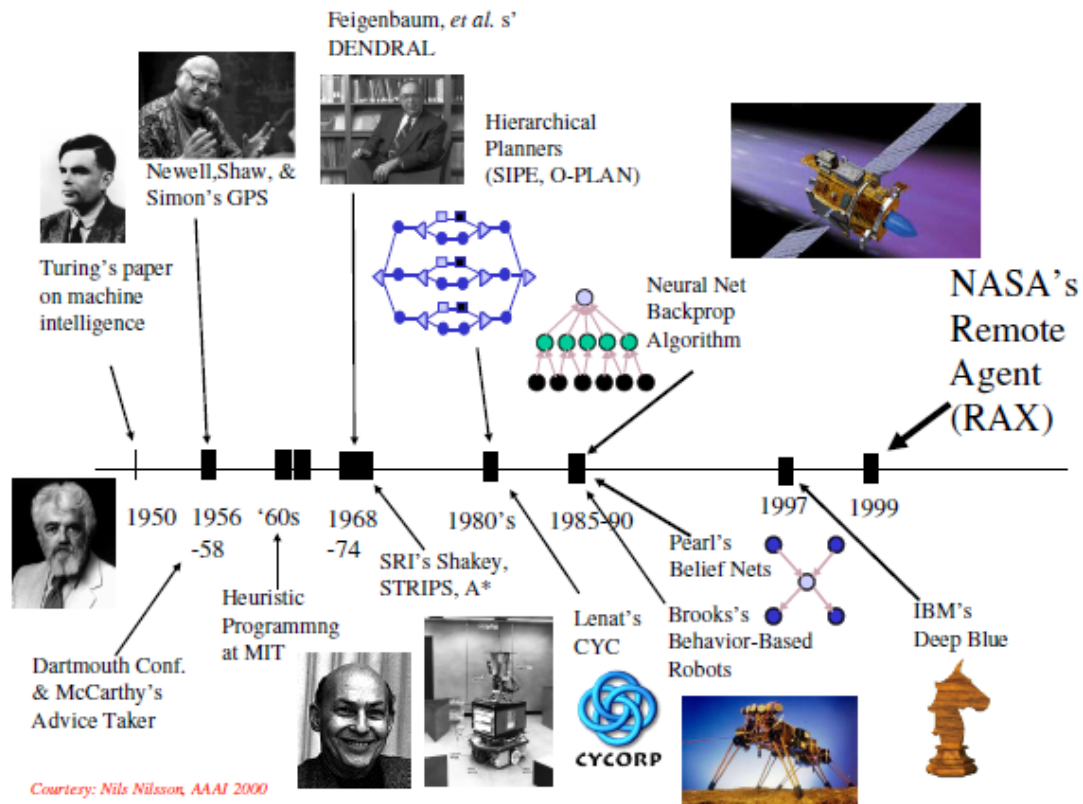
Progreso en el desarrollo de Sistemas Inteligentes



DENDRAL

1967: Dendral program (Edward Feigenbaum et al. at Stanford University) demonstrated to interpret mass spectra on organic chemical compounds. First successful **knowledge-based** program for scientific reasoning

Progreso en el desarrollo de Sistemas Inteligentes

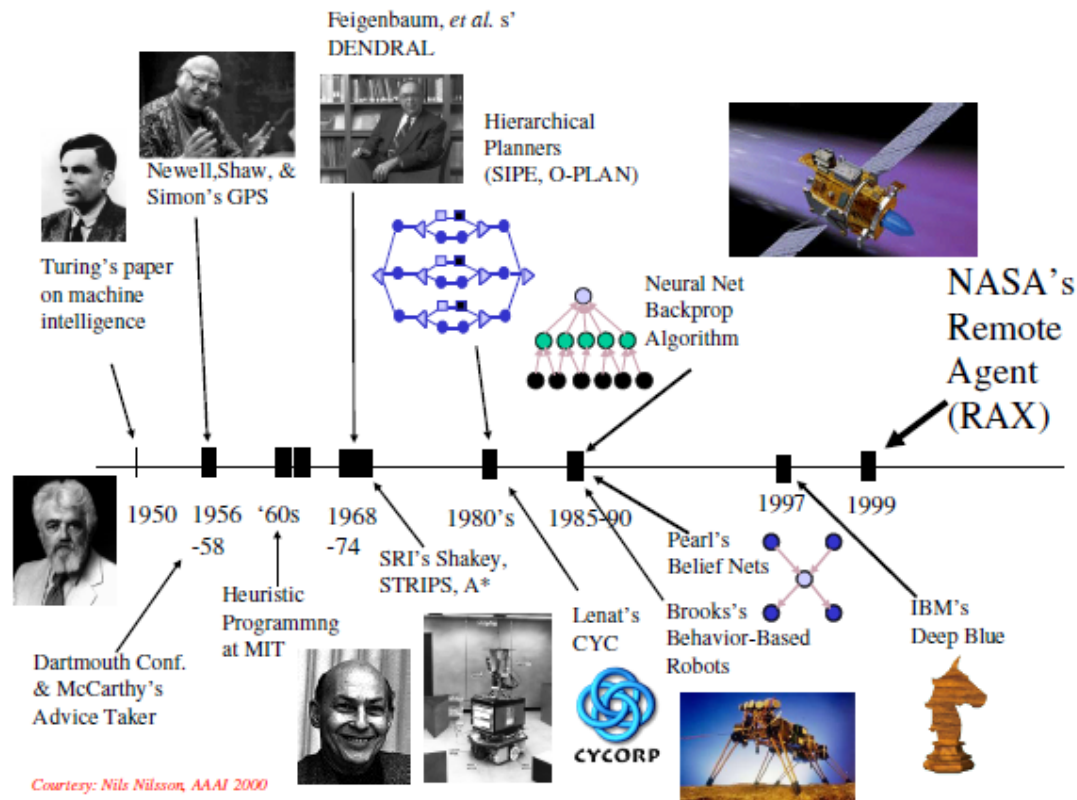


Stanford Research Institute (SRI)

1969: Shakey the Robot, demonstrated combining animal locomotion, perception and problem solving



Progreso en el desarrollo de Sistemas Inteligentes

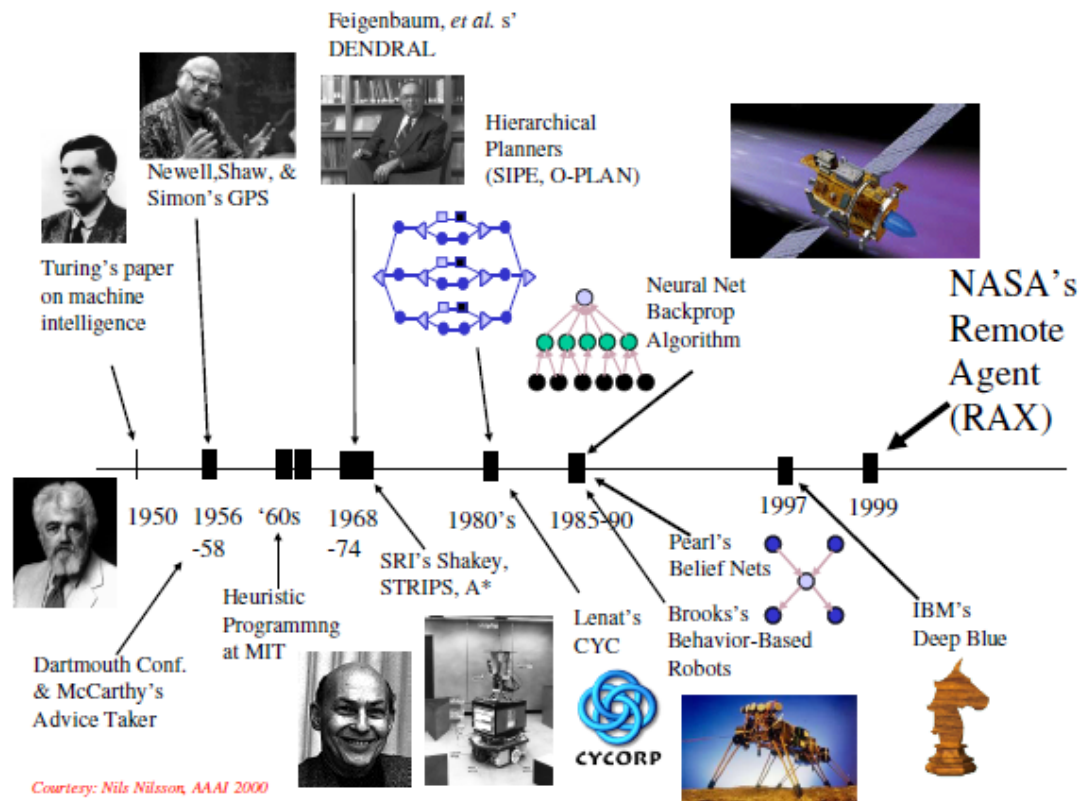


A*

1968: P. E. Hart, N. J. Nilsson, B. Raphael. *A formal basis for the heuristic determination of minimum cost paths*. IEEE Trans. Syst. Sci. Cybernet., volume 4, number 2, pp 100-107.

1972: P. E. Hart, N. J. Nilsson, B. Raphael. *Correction to a formal basis for the heuristic determination of minimum cost paths*. SIGART Newsletter, volume 37, number 28, p. 9.

Progreso en el desarrollo de Sistemas Inteligentes

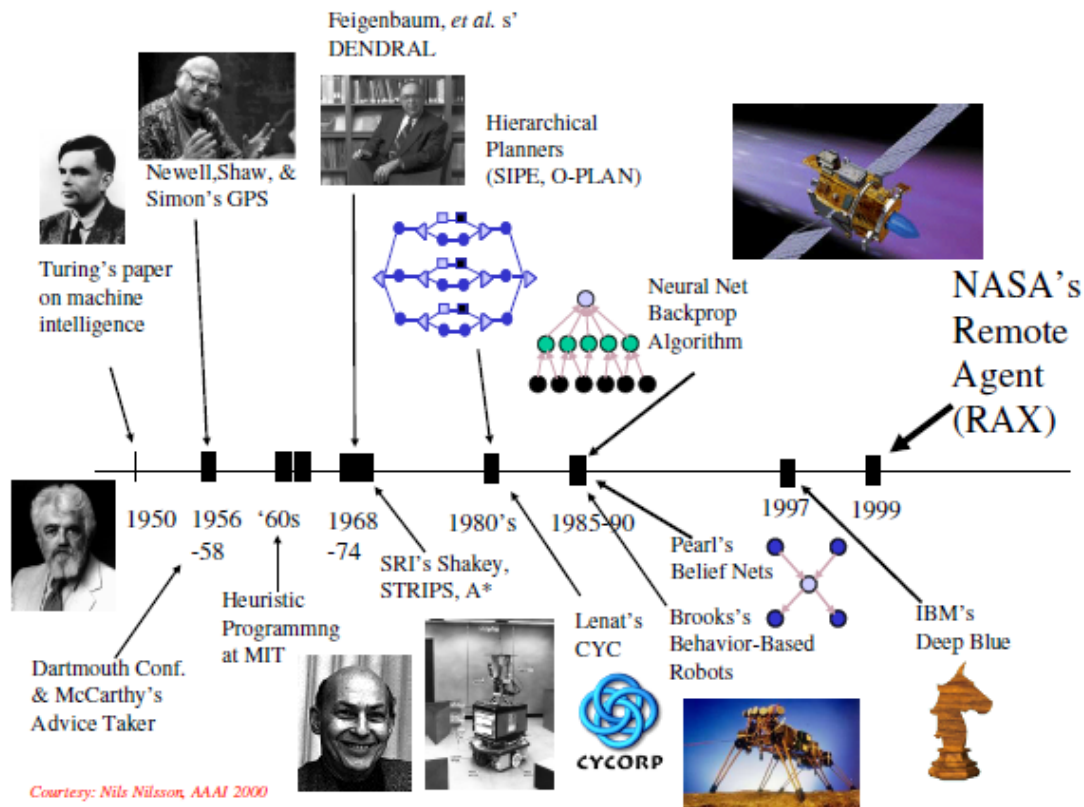


STRIPS

1971: STRIPS (STanford Research Institute Problem Solver) is an **automated planner** invented by Richard Fikes and Nils Nilsson

1974: Earl Sacerdoti developed one of the first planning programs, ABSTRIPS, and developed techniques of **hierarchical planning**.

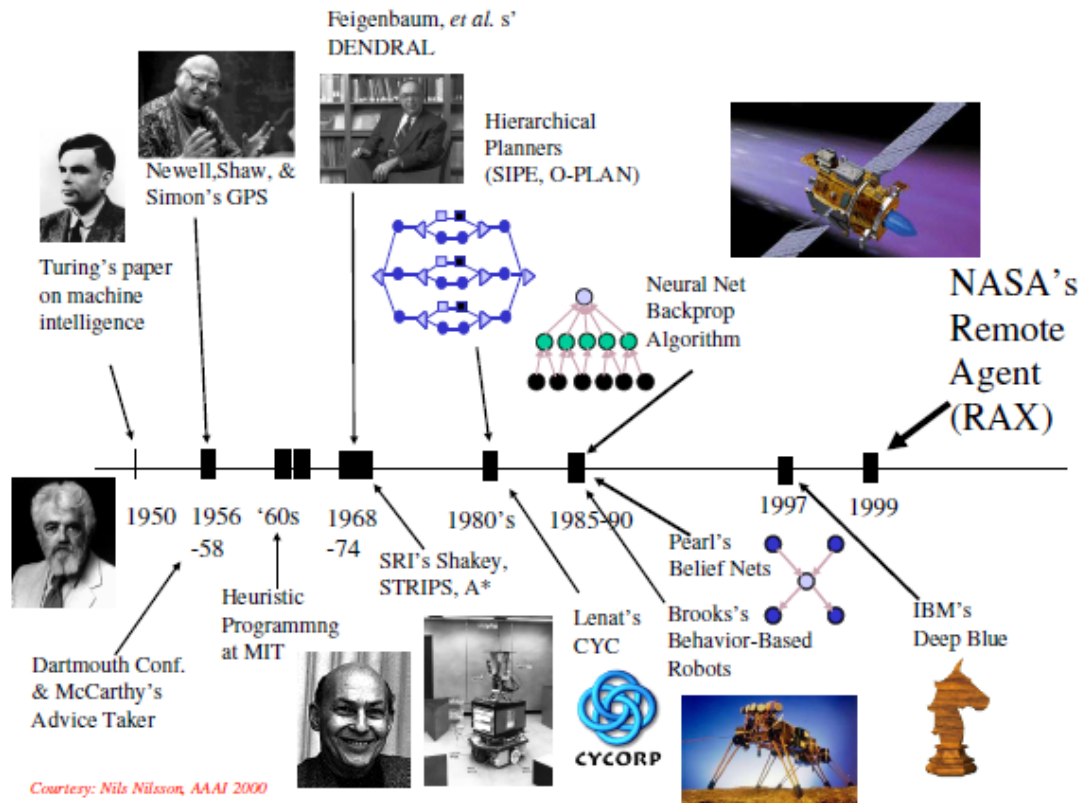
Progreso en el desarrollo de Sistemas Inteligentes




Hierarchical Planning

1983: O-PLAN has been used for a range of practical and research tasks. It was developed in 1983 and still runs as a planning service over the web (www.aiai.ed.ac.uk/project/oplan/)

Progreso en el desarrollo de Sistemas Inteligentes



Cycorp

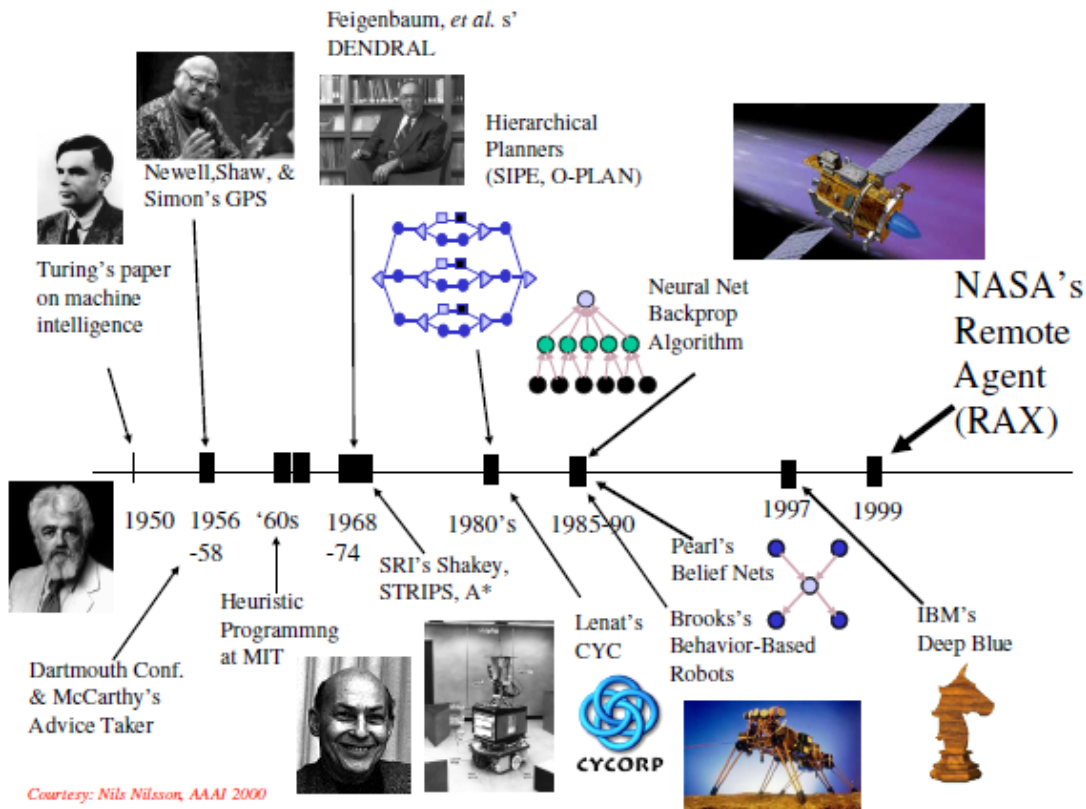


1984: Cycorp, Inc., based in Austin, Texas was founded by computer science professor at Stanford Douglas Lenat (www.cyc.com)

"Once you have a truly massive amount of information integrated as knowledge, then the human-software system will be superhuman, in the same sense that mankind with writing is superhuman compared to mankind before writing."

Doug Lenat, June 21, 2001

Progreso en el desarrollo de Sistemas Inteligentes



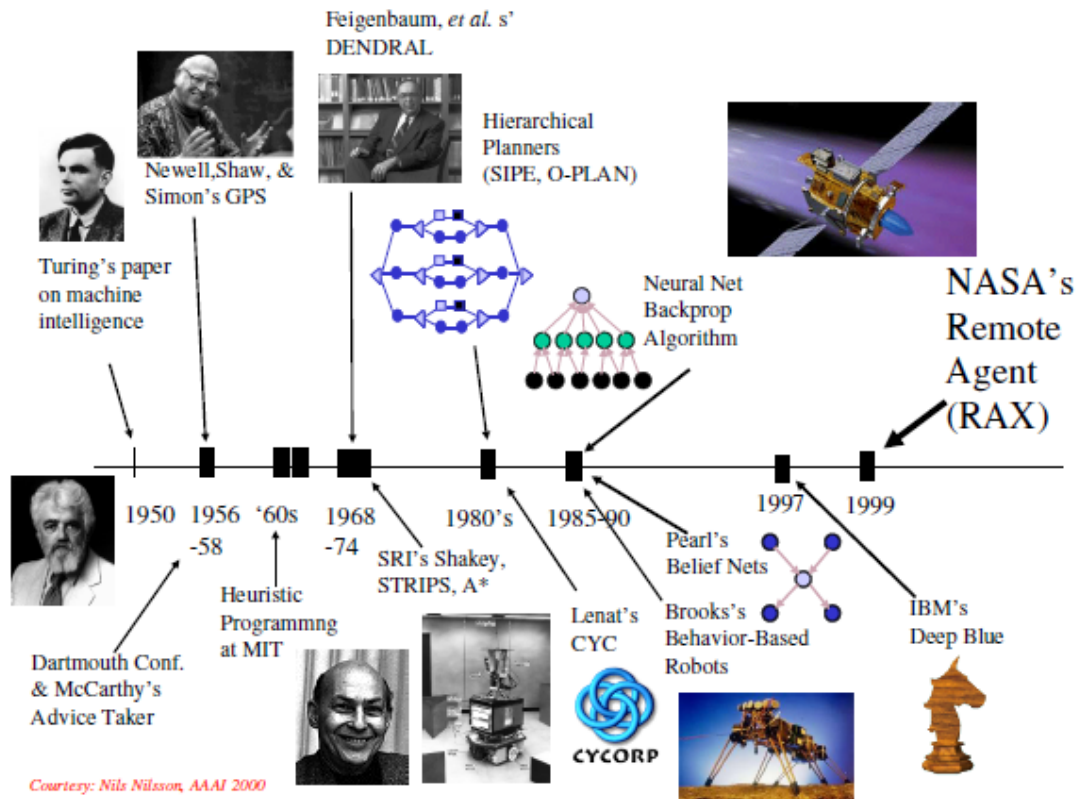
Neural Networks

1969: Marvin Minsky and Seymour Papert publish **Perceptrons**, demonstrating previously unrecognized limits of a simple form of **neural nets**

Mid 80's: Neural Networks become widely used with the **Backpropagation algorithm** —first described by Paul Werbos in 1974.

1989: Dean Pomerleau at CMU creates ALVINN: an Autonomous Land Vehicle in a Neural Network.

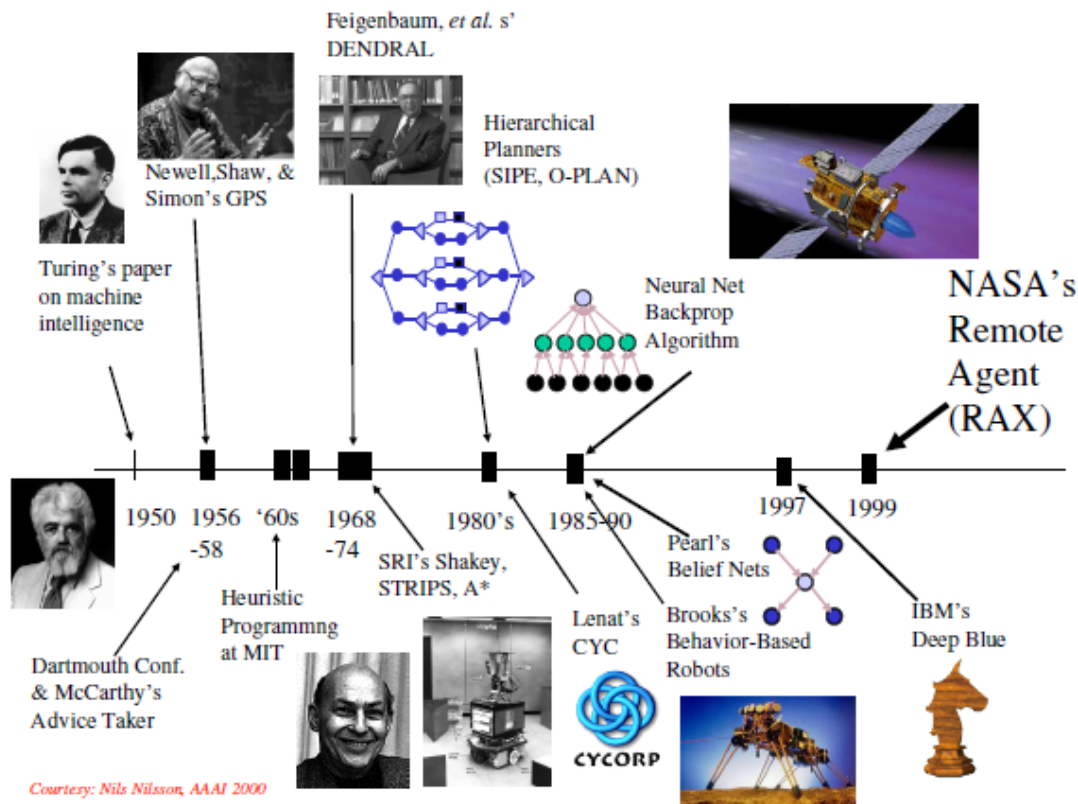
Progreso en el desarrollo de Sistemas Inteligentes



Robotics

1987: Rodney Brooks introduced the subsumption architecture and **behavior-based robotics** as a more minimalist modular model of natural intelligence.

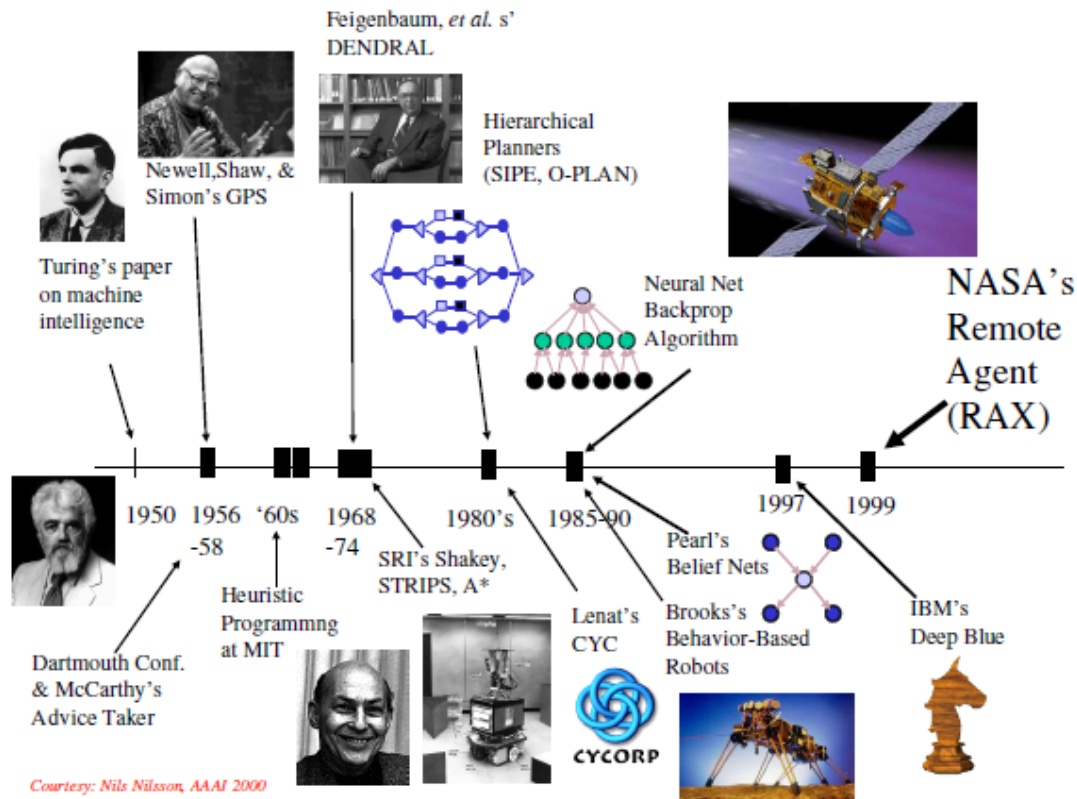
Progreso en el desarrollo de Sistemas Inteligentes



Belief Networks

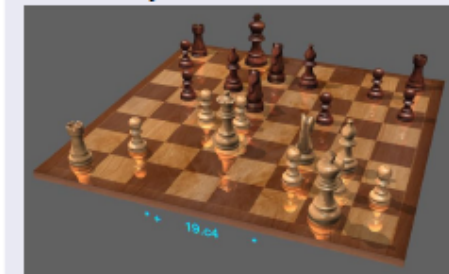
1988: Judea Pearl introduced **belief networks**, which are causal networks whose links are labeled with **probabilities**. The word belief is an important qualifier because *all the representations used in AI represent somebody's best guess or belief about causal influences rather than the ultimate facts of causation*

Progreso en el desarrollo de Sistemas Inteligentes

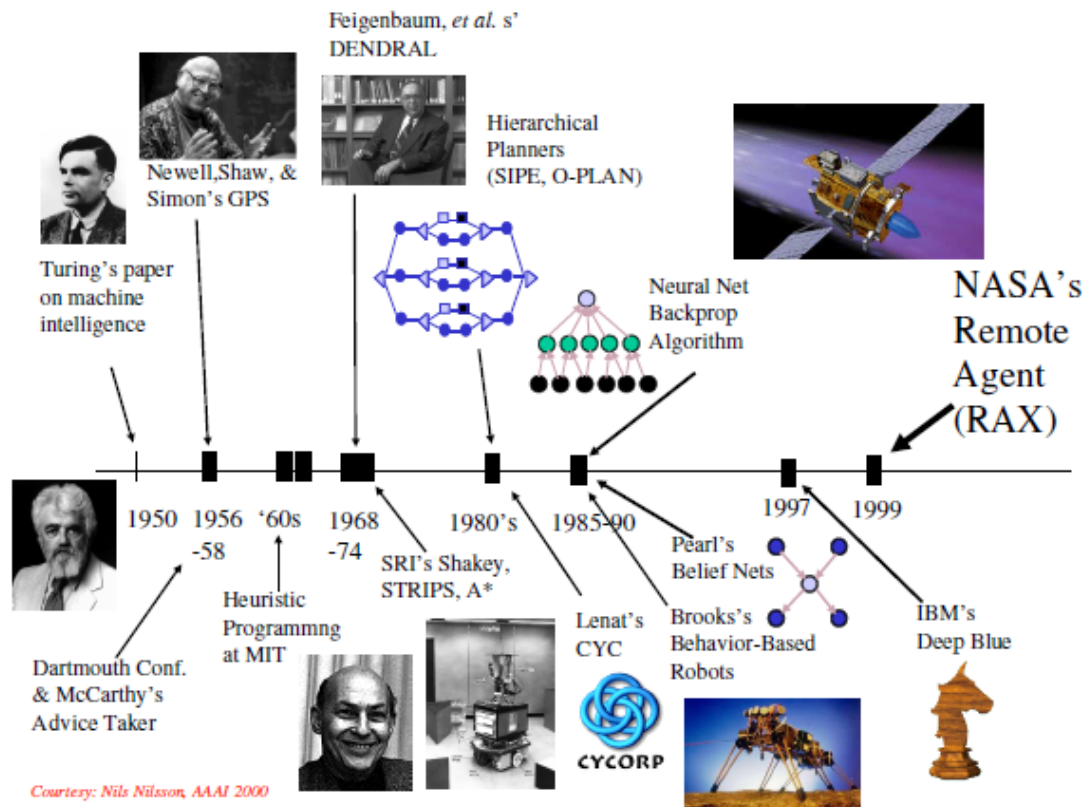


Deep Blue (IBM)

1997: The Deep Blue chess program (IBM) beats the world chess champion, Garry Kasparov, in a widely followed match.



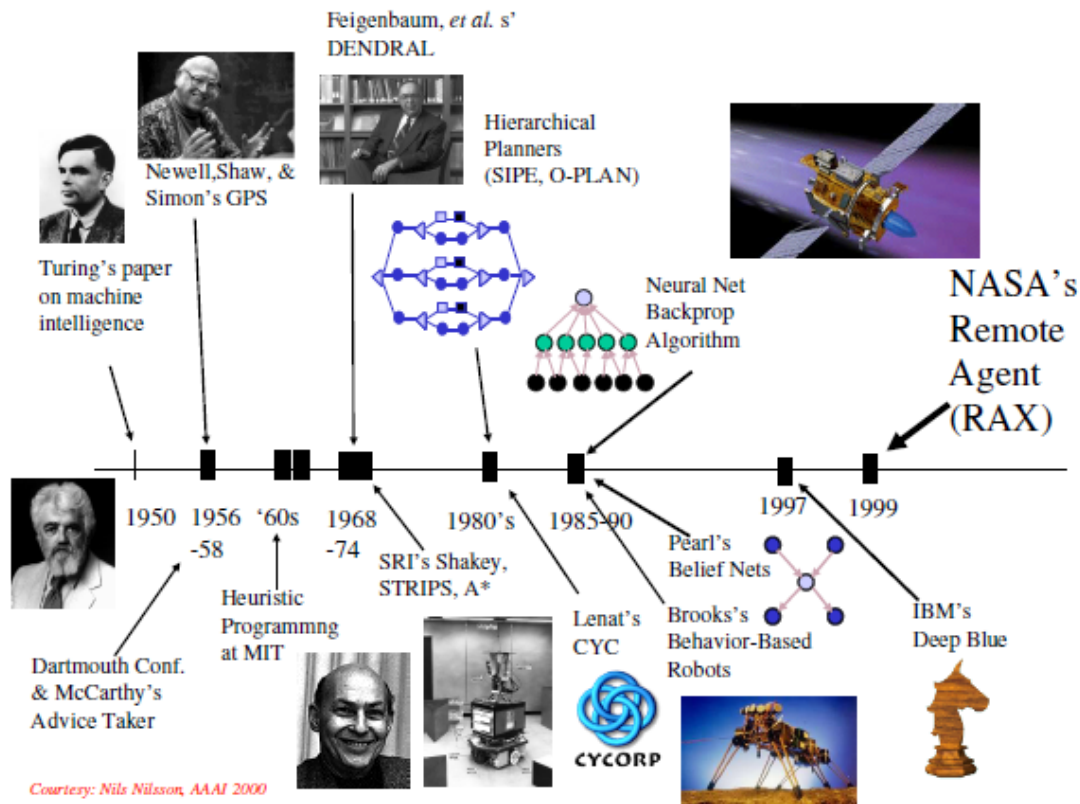
Progreso en el desarrollo de Sistemas Inteligentes



RAX

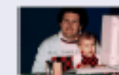
1999: Kanna Rajan was one of the principals of the Remote Agent Experiment (RAX) which designed, built, tested and flew the first AI based closed-loop control system on a spacecraft. The RAX was the co-winner of NASA's 1999 Software of the Year, the agency's highest technical award.

Progreso en el desarrollo de Sistemas Inteligentes



Chinook

2007: Computer Checkers Program Is Invincible. "Chinook [...] reduces checkers to the level of tic-tac-toe [...] it is the most complex game that has been solved to date" (Science - July, 19, 2007)



Progreso en el desarrollo de Sistemas Inteligentes

- AlphaZero: Un algoritmo de Inteligencia Artificial que aprende a ganar en tres juegos de estrategia diferentes: Ajedrez, Shogi y Go.
- Deep Learning: Traducción automática y clasificación de objetos en imágenes.

Construcción de Sistemas Inteligentes

- La Inteligencia Artificial es una rama de la Informática que estudia y resuelve problemas situados en la frontera de la misma.
- Se basa en dos ideas fundamentales:
 - Representación del conocimiento explícita y declarativa
 - Resolución de problemas (heurística)

MCA. Modelos de Computación Avanzada.

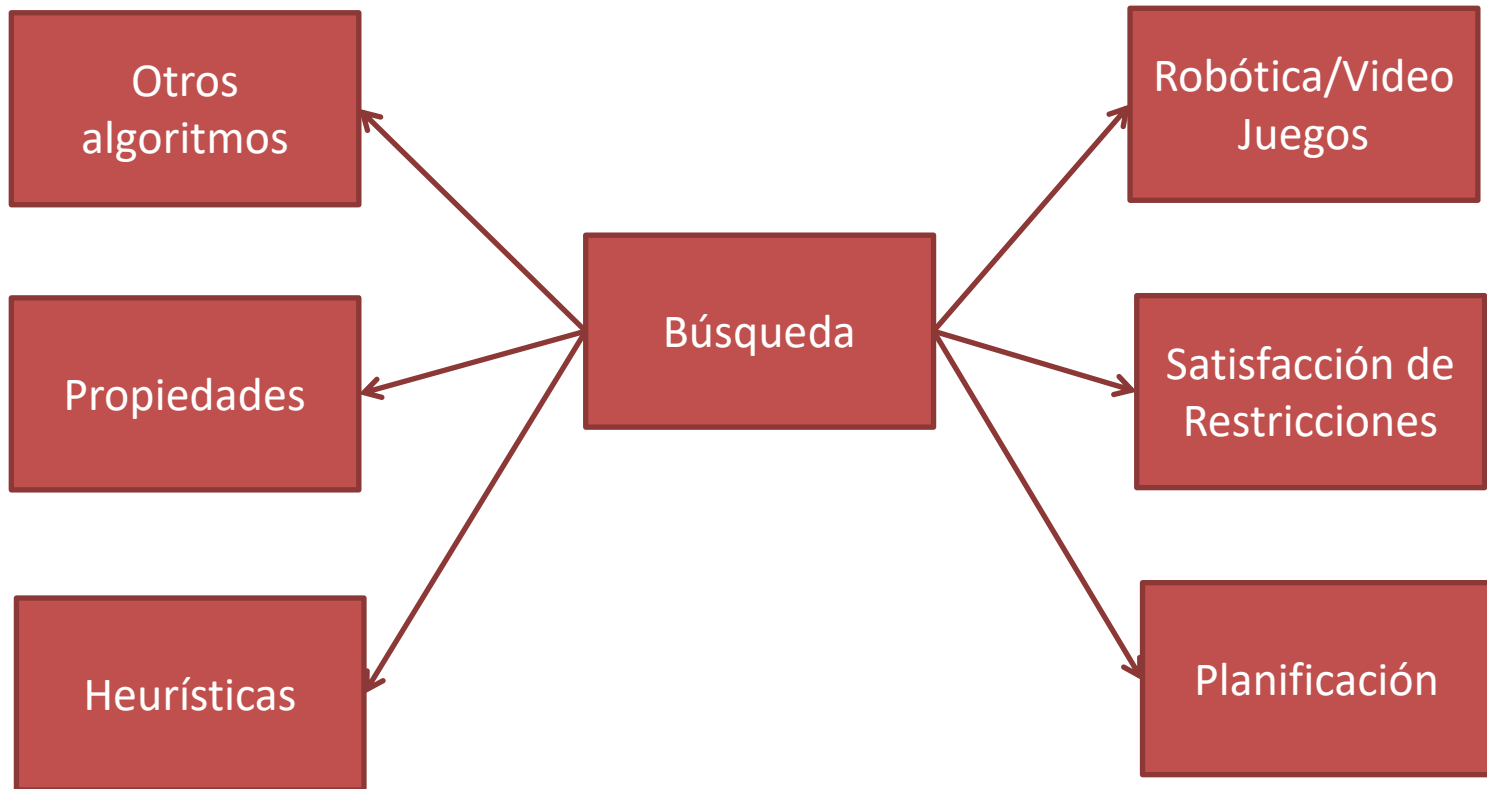
MH. Metaheurísticas.

TSI. Técnicas de los Sistemas Inteligentes.

IC. Ingeniería del Conocimiento.

AA. Aprendizaje Automático.

Búsqueda



Búsqueda heurística y propiedades

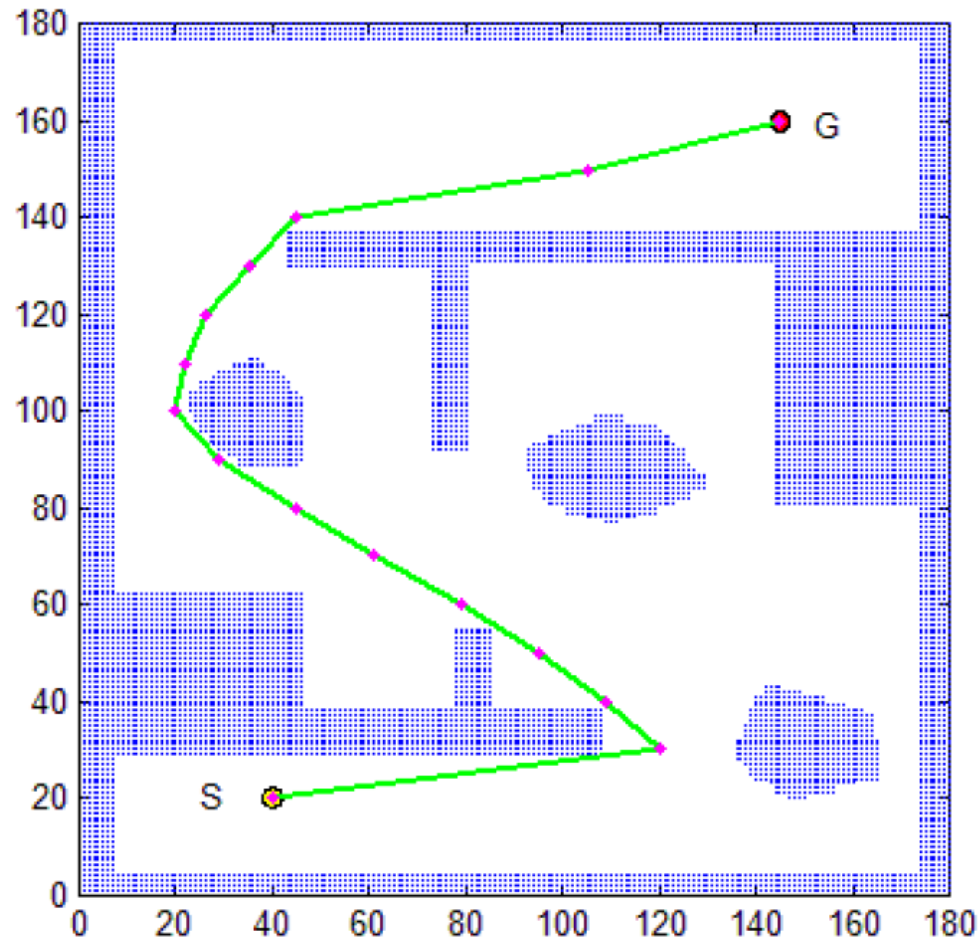
- Heurística y tipos de problemas
- Búsqueda primero el mejor
- Descomposición de problemas
- Propiedades de los métodos heurísticos
- Heurísticas a través de modelos simplificados

Heurística y tipos de problemas

Las **heurísticas** son criterios, métodos o principios para decidir cuál de entre varias acciones promete ser la mejor para alcanzar una determinada meta

- El problema de las ocho reinas
- El 8-puzzle
- El problema del mapa de carreteras
- El problema del viajante de comercio
- El problema de la moneda falsa

Planificación de caminos

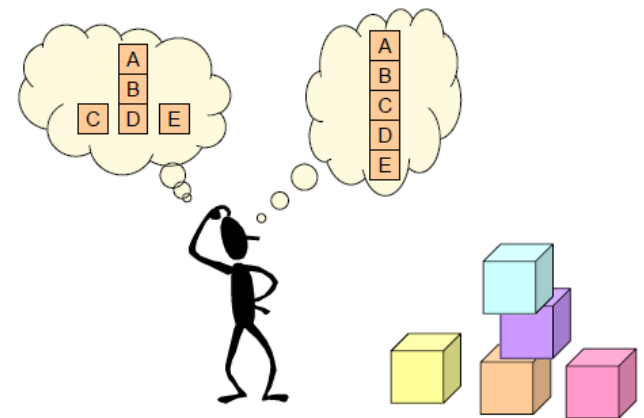


Planificación de caminos



Búsqueda primero el mejor

- Algoritmo A*
- Modelos más generales
- Modelos con poda
- Algunas alternativas al A*
- Planificación de caminos



Búsqueda con A*

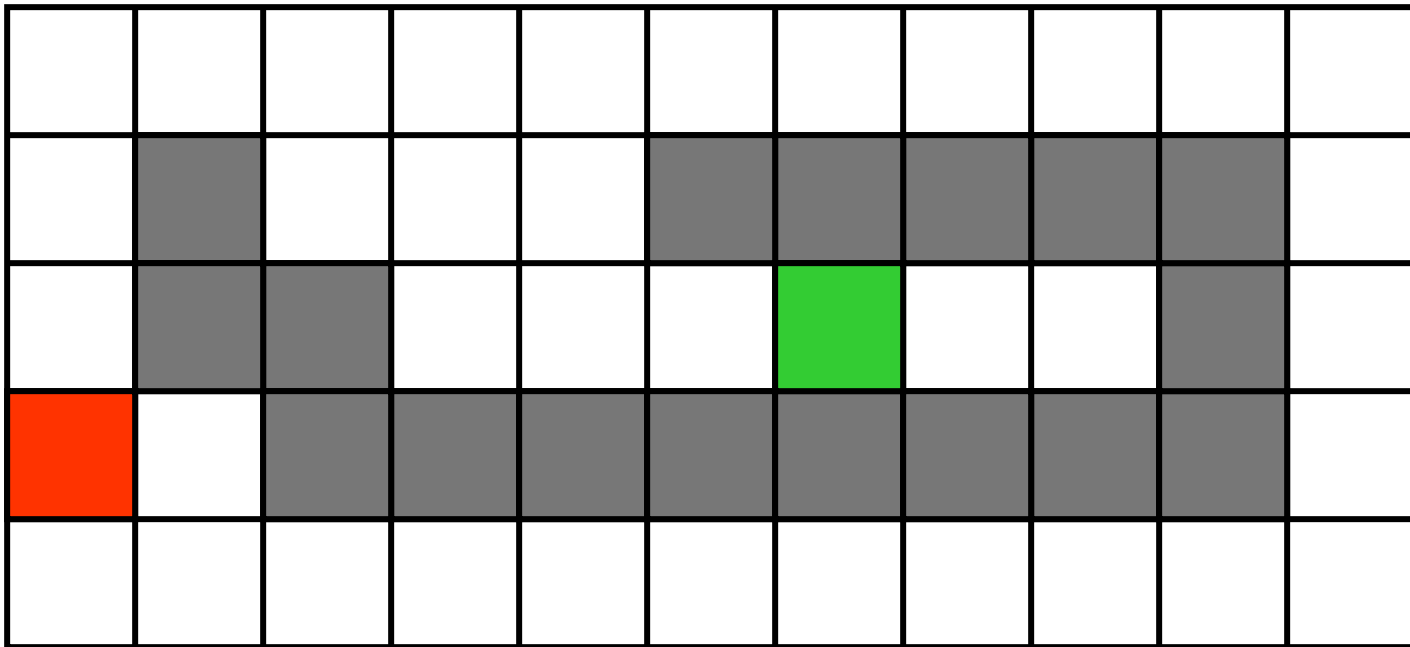
- Es el método más conocido de búsqueda por el mejor nodo.
- Idea: evitar explorar caminos que ya sabemos que no son interesantes por su costo.
- Función de evaluación $f(n)=g(n) + h(n) \rightarrow A^*$
 - $g(n)$ el costo (actual) .
 - $h(n)$ el costo estimado del nodo al objetivo.
 - $f(n)$ el costo total estimado al objetivo obligado a pasar por el nodo.

Algoritmo A* (Hart, Nilsson y Rafael, 1968)

- ABIERTOS contiene el nodo inicial, CERRADOS esta vacío
- Comienza un ciclo que se repite hasta que se encuentra solución o hasta que ABIERTOS queda vacío
 - Seleccionar el mejor nodo de ABIERTOS
 - Si es un nodo objetivo terminar
 - En otro caso se expande dicho nodo
 - Para cada uno de los nodos sucesores
 - Si está en ABIERTOS insertarlo manteniendo la información del mejor padre
 - Si está en CERRADOS insertarlo manteniendo la información del mejor padre y actualizar la información de los descendientes
 - En otro caso, insertarlo como un nodo nuevo

Planificación de caminos

- $f(N) = h(N) \rightarrow$ Búsqueda primero el mejor greedy



Planificación de caminos

- $f(N) = h(N)$, con $h(N)$ = distancia manhattan al objetivo

8	7	6	5	4	3	2	3	4	5	6
7		5	4	3						5
6			3	2	1	0	1	2		4
7	6									5
8	7	6	5	4	3	2	3	4	5	6

Planificación de caminos

- ¿Qué ocurre?

8	7	6	5	4	3	2	3	4	5	6
7		5	4	3						5
6			3	2	1	0	1	2		4
7	6									5
8	7	6	5	4	3	2	3	4	5	6

Planificación de caminos

- $f(N) = g(N) + h(N)$, con $h(N)$ = distancia manhattan al objetivo

8+3	7+4	6+3	5+6	4+7	3+8	2+9	3+10	4	5	6
7+2		5+6	4+7	3+8						5
6+1			3	2+9	1+10	0+11	1	2		4
7+0	6+1									5
8+1	7+2	6+3	5+4	4+5	3+6	2+7	3+8	4	5	6

Modelos más generales

- Algoritmos de agendas

Modelos con poda

- Búsqueda dirigida
- Algoritmo A^* con memoria acotada

Algunas alternativas al A*

- Descenso iterativo A*
- Búsqueda primero el mejor recursiva
- Búsqueda por franjas

IDA* (Korf, 1985)

Procedimiento IDA* (Estado-inicial Estado-final)

EXITO=Falso

$$\eta = h(s)$$

Mientras que EXITO=Falso

 EXITO=Profundidad (Estado-inicial, η)

$$\eta = \min_{i=1,n} \{f(i)\} = \min_{i=1,n} \{g(i) + h(i)\}$$

Solución=camino desde nodo del Estado-inicial
 al Estado-final por los punteros

Profundidad (Estado-inicial, η)

 Expande todos los nodos cuyo coste $f(n)$ no excede η

Características

- **Completitud:** es completo
- **Admisibilidad:** es admisible
- **Eficiencia:** la complejidad de tiempo es exponencial, pero la complejidad de espacio es lineal en la profundidad del árbol de búsqueda
- Aunque pudiera parecer lo contrario, el numero de re-expansiones es solo mayor en un pequeño factor que el número de expansiones de los algoritmos de primero el mejor
- Fue el primer algoritmo que resolvió óptimamente 100 casos generados aleatoriamente en el 15-Puzzle

Búsqueda por franjas

- Fringe Search (Björnsson et al. 2005)
 - Estados repetidos: uso de una tabla
 - Revisita nodos: uso de dos listas

Búsqueda por franjas

IDA*	Fringe Search	f_{limit}
		4
		5
		6

Maneja dos listas:

- lista now
- lista later

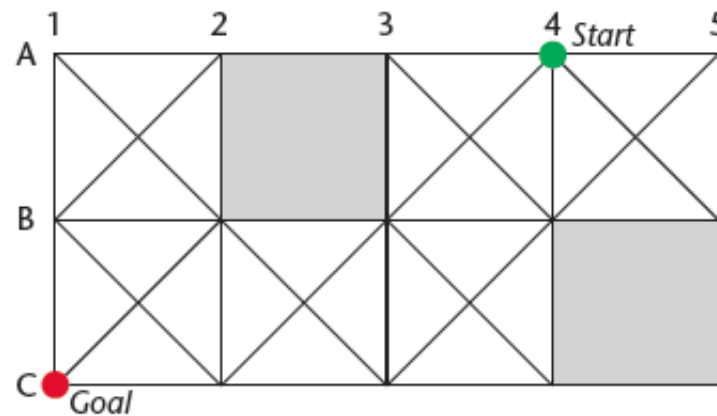
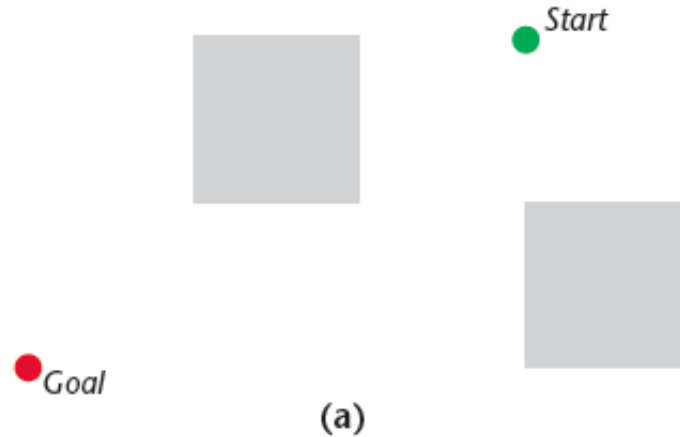
Entorno gráfico

- <https://qiao.github.io/PathFinding.js/visual/>

Algoritmos de búsqueda para planificación de caminos con otro modelo de representación

- Modelo de celdas
- Celdas completamente ocupadas o libres
- También se suelen usar las esquinas de las celdas como puntos del proceso de búsqueda
- Vértices con línea de visión
- $c(s,s')$ costo de la línea recta entre dos vértices
- Conjunto de vecinos visibles de un vértice

Planificación de caminos

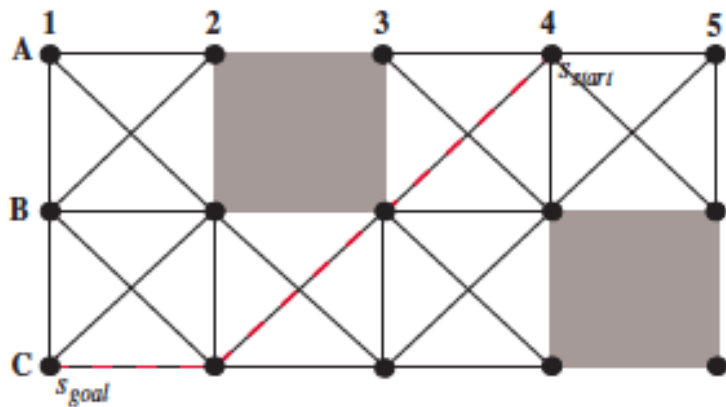


Algoritmo A*

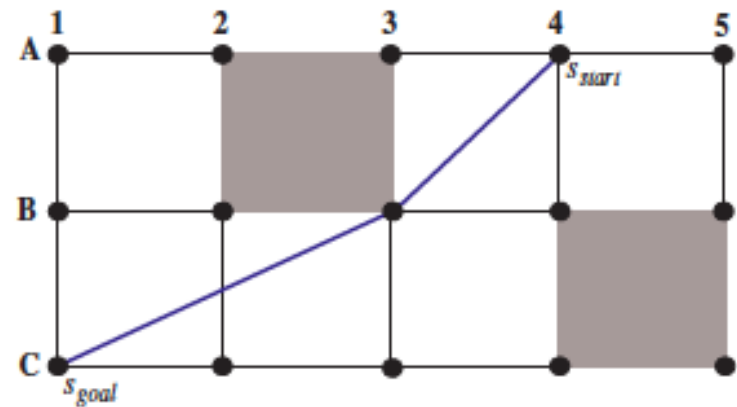
```
1 Main()
2    $g(s_{start}) := 0;$ 
3    $parent(s_{start}) := s_{start};$ 
4    $open := \emptyset;$ 
5    $open.Insert(s_{start}, g(s_{start}) + h(s_{start}));$ 
6    $closed := \emptyset;$ 
7   while  $open \neq \emptyset$  do
8      $s := open.Pop();$ 
9     if  $s = s_{goal}$  then
10       return "path found";
11      $closed := closed \cup \{s\};$ 
12     /* The following line is executed only by AP Theta*. */
13     [UpdateBounds(s)];
14     foreach  $s' \in neighbors_{vis}(s)$  do
15       if  $s' \notin closed$  then
16         if  $s' \notin open$  then
17            $g(s') := \infty;$ 
18            $parent(s') := NULL;$ 
19           UpdateVertex(s, s');
20   return "no path found";
21 end

22 UpdateVertex(s,s')
23   if  $g(s) + c(s, s') < g(s')$  then
24      $g(s') := g(s) + c(s, s');$ 
25      $parent(s') := s;$ 
26     if  $s' \in open$  then
27        $open.Remove(s');$ 
28      $open.Insert(s', g(s') + h(s'));$ 
29 end
```

A* sobre celdas

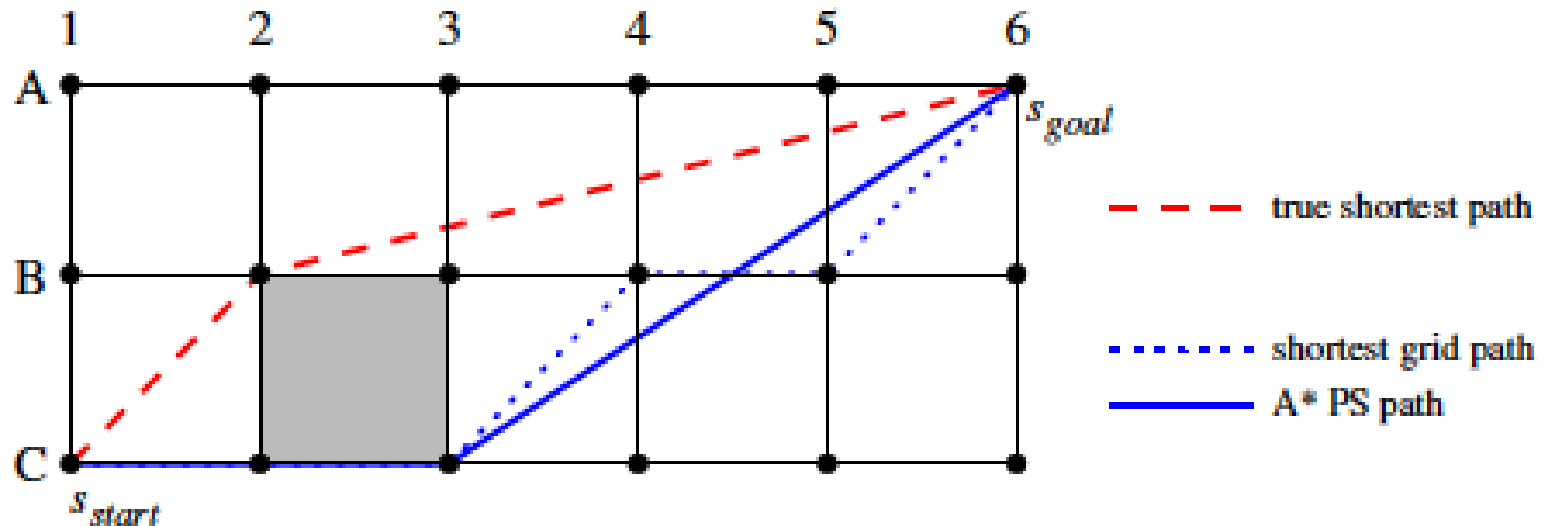


(a) Grid path

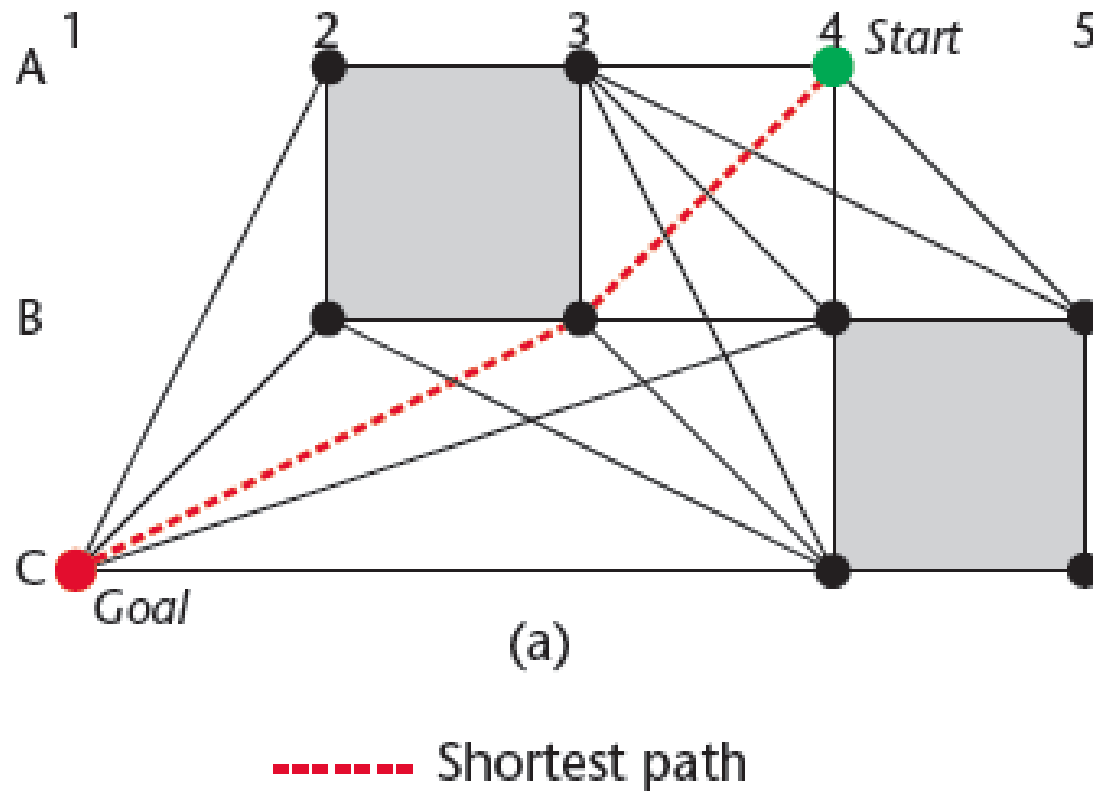


(b) True shortest path

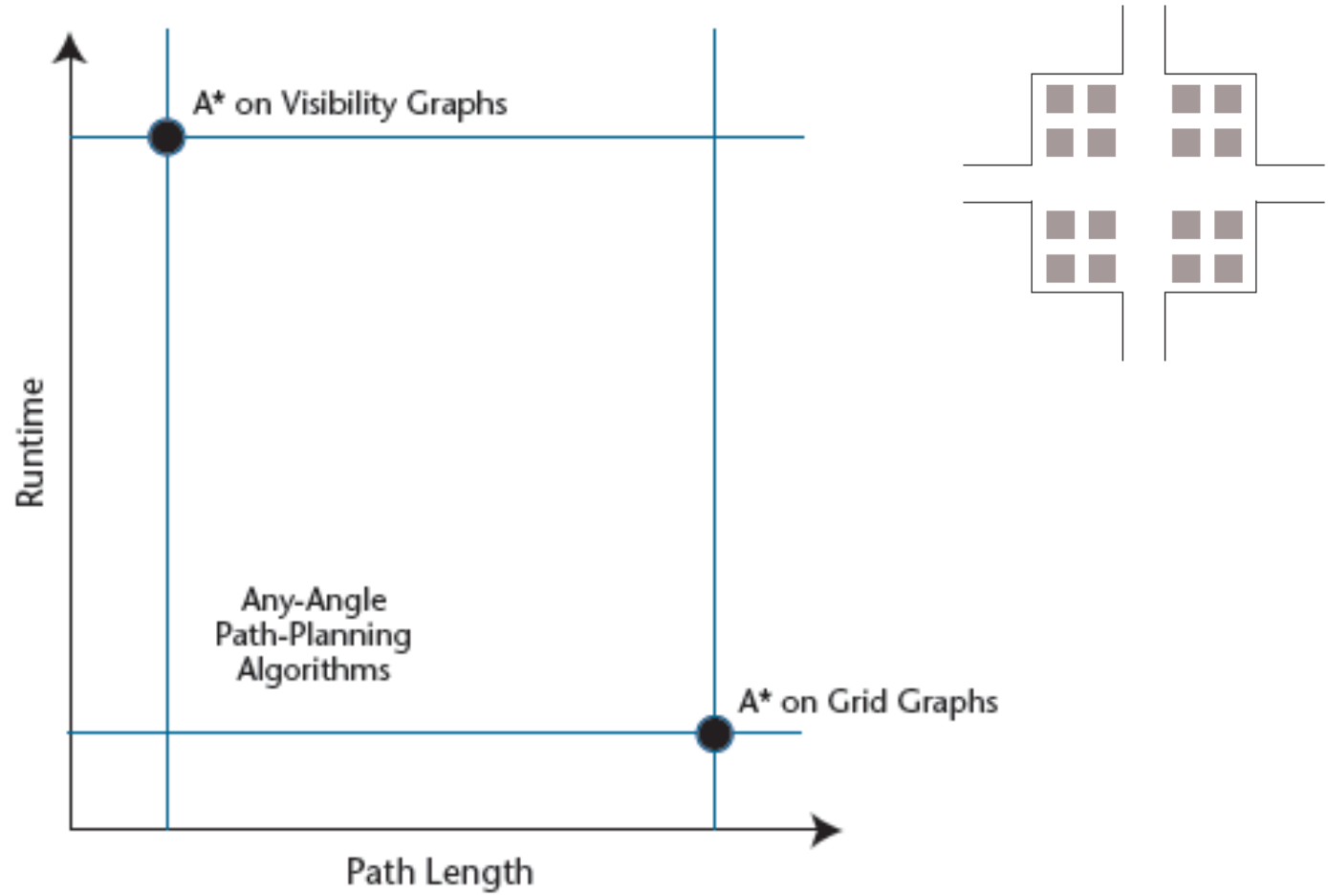
A* con caminos suavizados a posteriori



A* sobre grafos de visibilidad



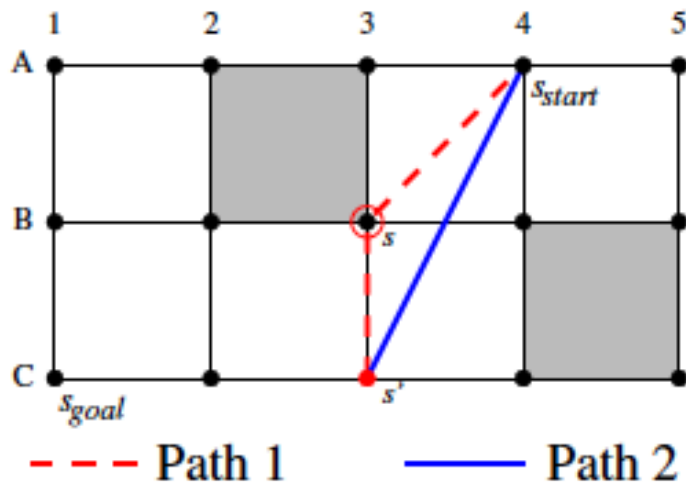
A* sobre grafos de visibilidad



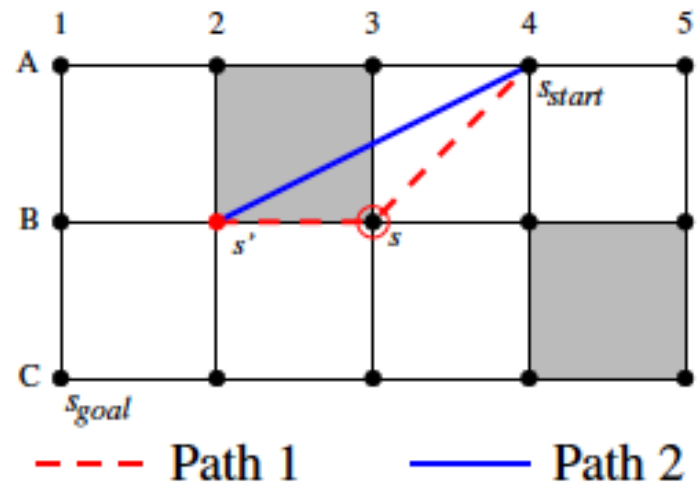
Algoritmo Theta* básico

- Nodo de inicio \rightarrow nodo $s \rightarrow$ nodo s'
 - Camino 1: Considera el coste desde el nodo de inicio al nodo s más el coste del nodo s a su sucesor s' .
 - Camino 2: Considera el coste del nodo de inicio al padre(s) más el coste del padre(s) a s' en línea recta.

Algoritmo Theta* básico

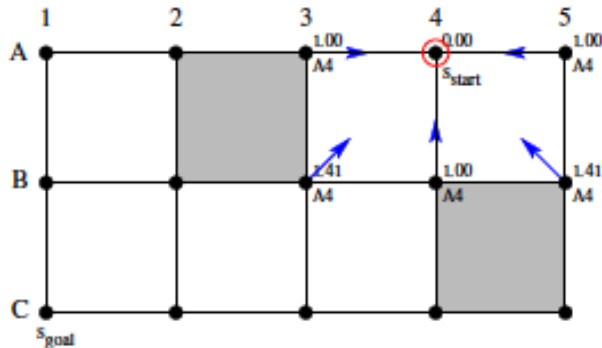


(a) Path 2 is unblocked

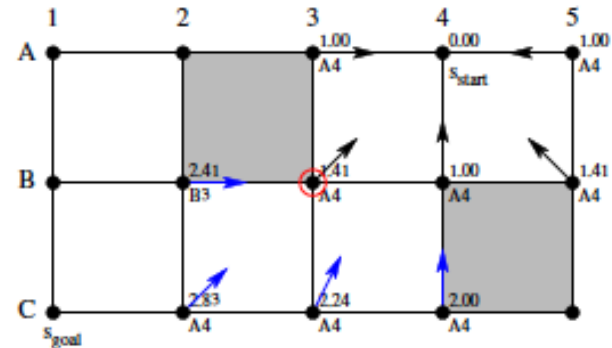


(b) Path 2 is blocked

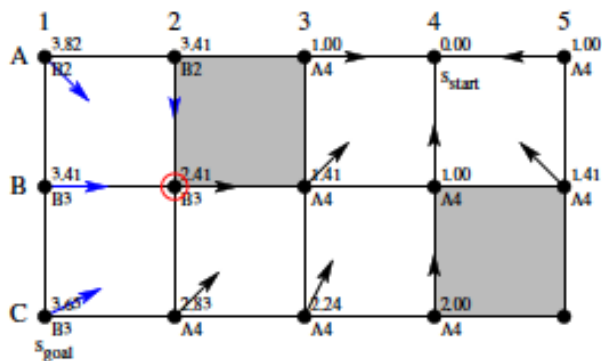
Ejemplo de Theta* básico



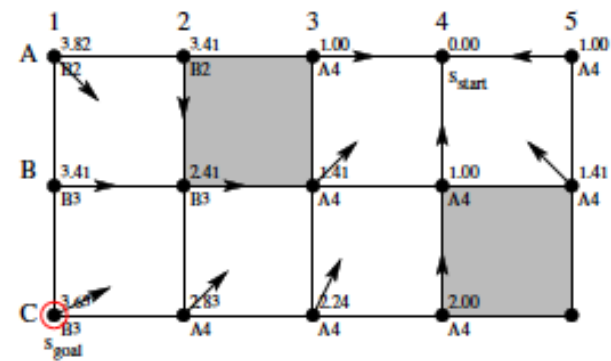
(a)



(b)



(c)



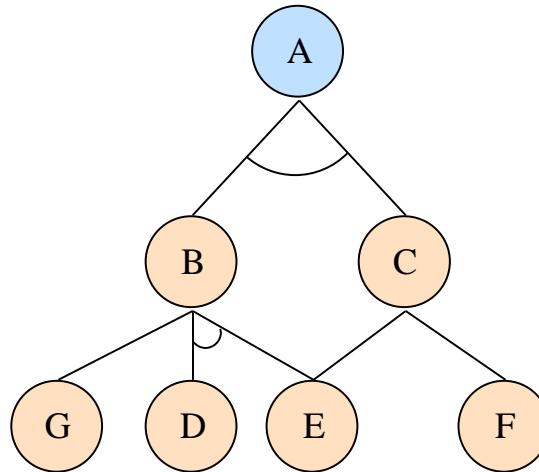
(d)

Descomposición de problemas

- Algoritmo Y/O*

Grafo Y/O

- **Grafo Y/O:** Combinación de nodos Y y nodos O que indican el orden de consecución de tareas a realizar para alcanzar el objetivo.



- En el cálculo proposicional, la expresión del grafo Y/O anterior correspondiente sería de la siguiente forma:

$$\mathbf{B \cdot C \rightarrow A; G + D \cdot E \rightarrow B; E + F \rightarrow C}$$

Grafo Y/O

- **Para resolver un grafo Y/O**, cada nodo se resuelve de la siguiente manera:
 - Si es un nodo Y: Resolver todos sus hijos. Combinar la solución y solucionar el nodo. Devolver su solución.
 - Si es un nodo O: Resolver un hijo y ver si devuelve solución. En caso contrario, resolver el siguiente hijo, etc. Cuando ya esté resuelto algún hijo, combinar la solución en el nodo y devolverla.
 - Si es un nodo terminal: Resolver subproblema asociado y devolverla.
- **Mejora:** Para seleccionar el orden de resolución de nodos hijos, se puede utilizar alguna medida de estimación del coste de resolución.

Algoritmo Y/O*

- GRAFO contiene el nodo de inicio.
- Comienza un ciclo que se repite hasta que el nodo inicial quede resuelto o hasta que supere un valor MAXIMO
 - Trazar el mejor camino actual desde inicio
 - Seleccionar un nodo
 - Generar los sucesores del nodo e incluirlos de forma adecuada en el GRAFO
 - Propagar la información obtenida hacia arriba en el GRAFO

Propiedades de los métodos heurísticos

- ¿Encuentra A^* solución?
- ¿Encuentra A^* la solución óptima?
- Cuando hay dos heurísticas para un problema
¿Cuál es la mejor?
- ¿Es posible encontrar heurísticas que hagan
más eficiente a A^* ?

Dominancia

- Heurística h_2 más informada que h_1 si ambas son admisibles y $h_2 > h_1$
- Teorema: Si A^* usando h_2 esta más informado que A^* usando h_1 , entonces el primero domina al segundo.
- ¿Qué ocurre si $h_2 \geq h_1$?
- ¿Cuál es el interés de usar? $f(n)=g(n)+wh(n)$?

Propiedades

- Un algoritmo A^* guiado por una heurística monótona encuentra los caminos óptimos hacia todos los nodos expandidos, es decir,

$$g(n)=g^*(n) \forall n \in \text{CERRADOS}$$

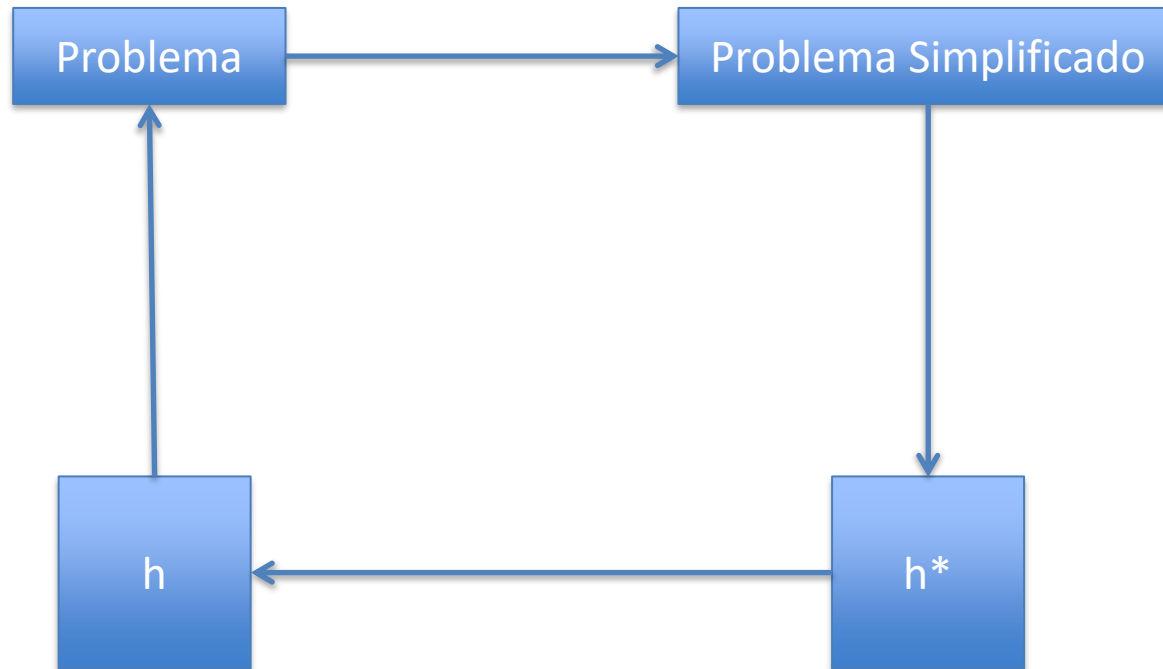
Heurísticas a través de modelos simplificados

- ¿Cómo se generan las heurísticas?
- ¿Cómo saber que tienen buenas propiedades?

Heurísticas a través de modelos simplificados

- Heurística mapa de carreteras
- Heurística 8-puzzle
- Heurística viajante de comercio
 - Ser grafo
 - Estar conectado
 - Ser de grado 2

Heurísticas a través de modelos simplificados



La heurística obtenida para el problema es el valor h^* del problema simplificado

Propiedades

- Teorema: toda heurística obtenida por un modelo simplificado es consistente

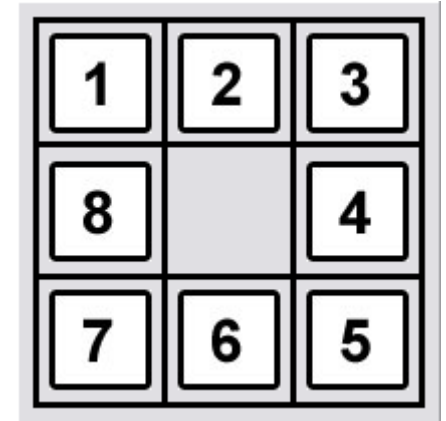
$$h^*(n) \leq c_s(n, n') + h^*(n')$$

$$c_s(n, n') \leq c(n, n')$$

$$h(n) \leq c(n, n') + h(n')$$

Proceso sistemático

- Descripción formal del 8-puzzle
- Predicados
 - $\text{SOBRE}(X,Y)$
 - $\text{LIBRE}(Y)$
 - $\text{ADY}(Y,Z)$
- Precondiciones del operador $\text{MOVER}(X,Y,Z)$
 $\text{SOBRE}(X,Y) \wedge \text{LIBRE}(Y) \wedge \text{ADY}(Y,Z)$



Proceso sistemático

- Modelo 1: $\text{SOBRE}(X,Y)$
- Modelo 2: $\text{SOBRE}(X,Y) \wedge \text{ADY}(Y,Z)$
- Modelo 3: $\text{SOBRE}(X,Y) \wedge \text{LIBRE}(Y)$
- Otros modelos
- Reflexión: En muchos casos los modelos simplificados corresponden a versiones descomponibles de problemas que no lo son directamente