

Esto no son apuntes pero **tiene un 10 asegurado** (y lo vas a disfrutar igual).

Abre la Cuenta NoCuenta con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Me interesa

1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en [ing.es](https://www.ing.es)



INTELIGENCIA ARTIFICIAL

CURSO 23/24

Consulta
condiciones aquí



do your thing

WUOLAH

¿Qué significa ser inteligente?

- La inteligencia es la capacidad de **ordenar los pensamientos y coordinarlos** con las acciones. La inteligencia no es una sola, sino que **existen tipos distintos**. (Howard Gardner, teoría de inteligencias múltiples)

Definición de IA

¥ Sistemas que piensan como humanos (**modelos cognitivos**)

- Es el funcionamiento de la mente humana
- Se intenta establecer una teoría sobre el funcionamiento de la mente con experimentación psicológica
- A partir de la teoría podemos establecer modelos computacionales con [redacted] a

¥ Sistemas que piensan racionalmente (leyes del **pensamiento**)

- Las leyes del pensamiento racional se fundamentan en la lógica
- La lógica formal está en la base de los programas inteligentes [redacted] icismo
- Se presentan dos **obstáculos**:
 - Es muy difícil formalizar el conocimiento
 - Hay un gran salto entre la capacidad teórica de la lógica y su realización práctica

¥ Sistemas que actúan como humanos (test de **Turing**)

- El objetivo es construir un sistema que pase por humano
- La interacción de programas con personas hace necesarias las máquinas que actúan como seres humanos
- Las tareas que de momento realizan mejor los humanos son:
 - Tareas de los **expertos** (detección de fallos, análisis científico, etc)
 - Trabajos de **vida diaria** (percepción, lenguaje natural, etc)
 - Tareas **formales** (juegos, matemáticas, etc)

¥ Sistemas que actúan racionalmente (**agentes racionales**)

- Un agente racional actúa de la **manera correcta** según la información que posee
- Parecido a los sistemas que actúan como humanos pero con una **visión más general**, no centrada en humanos

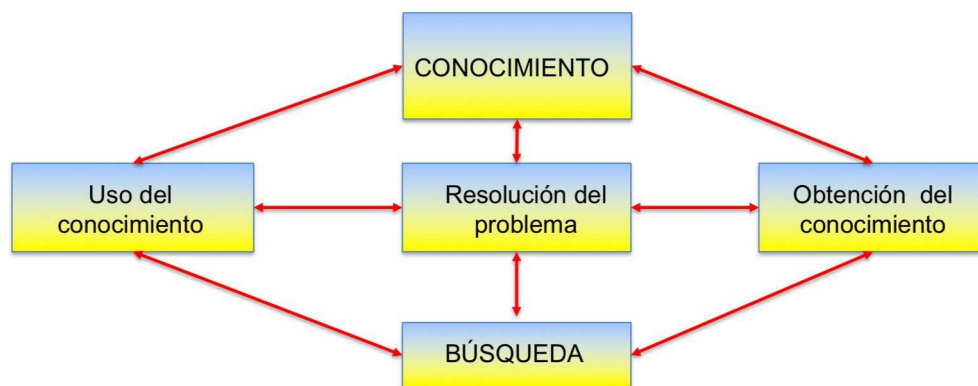
La inteligencia artificial es una rama de la informática que estudia y resuelve problemas situados en la frontera de la misma

La IA se basa en dos ideas fundamentales:

1. representaci—n del conocimiento expl'cita y declarativa
2. resoluci—n de problemas (**heur'stica**)

Tiene dos formas de resolver problemas:

1. mediante bœsqueda o uso del conocimiento disponible
2. mediante un proceso de obtener o aprender conocimiento



La inteligencia artificial plantea problemas filos—ficos complejos:

- ¥ ELIZA (test de turing)
- ¥ La habitaci—n china (Searle, 1980)

Bases de la IA

Econom'a (teor'a de la decisi—n, teor'a de juegos, investigaci—n operativa)

- ¥ Como debemos tomar decisiones que nos beneficien?
- ¥ Como debemos tomar decisiones en contra de la competencia?
- ¥ Como debemos tomar decisiones cuando el beneficio no es inmediato?

Neurociencia (neuronas, especializaci—n del cerebro)

- ¥ Como procesa la informaci—n el cerebro?

Psicolog'a (teor'as sobre la conducta, bases del comportamiento racional)

- ¥ Como piensan y actœan las personas?

Computaci—n

- ¥ Es necesario un mecanismo que soporte la IA (hardware)
- ¥ Son necesarias tambiœn herramientas para desarrollar programas de AI

Teor'a de Control y CibernŽtica

- ¥ Construcci—n de sistemas aut—nomos

LingŸ'stica

- ¥ Chomsky: gram'tica de la lengua
- ¥ LingŸ'stica computacional

Esto no son apuntes pero **tiene un 10 asegurado** (y lo vas a disfrutar igual).

Abre la Cuenta NoCuenta con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Me interesa

1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en [ing.es](#)



Historia de la IA

Periodo de Gestación (1943-1955): primeros modelos neuronales que simulan una neurona biológica (McCulloch y Pitts, 1943)

Nacimiento (1956): se acuña el nombre Inteligencia Artificial en la conferencia Dartmouth

Entusiasmo Inicial con grandes Expectativas (1952-1969): General/Geometry Problem Solver, Advice Taker, mundo de los bloques, hipótesis de sistema de símbolos físicos, etc

Dosis de Realidad (1966-1973): aparecen dificultades para resolver algunos problemas

Sistemas Expertos (1969-1986): primeros sistemas expertos, DENDRAL para reconocer moléculas, MYCIN para diagnóstico médico, LISP, etc

IA en la industria (1980-presente): control difuso, diseño de chips, interfaces hombre-máquina, algoritmos heurísticos, etc

1986-presente : nueva era redes neuronales
1987-presente: razonamiento probabilístico
2011-presente: big data & deep learning

Áreas de trabajo de la IA

- | | |
|-----------------------------------|---------------------------------------|
| ¥ Representación del conocimiento | ¥ Planificación de tareas |
| ¥ Resolución de problemas | ¥ Tratamiento del Lenguaje Natural |
| ¥ Búsqueda | ¥ Razonamiento Automático |
| | ¥ Sistemas Basados en el Conocimiento |
| | ¥ Percepción |
| | ¥ Aprendizaje Automático |
| | ¥ Agentes autónomos |

Otros Aspectos de la IA

- ¥ Actualmente se procesan gran **volumen de datos**, dicho procesamiento implica **procesado inteligente**.
- ¥ La IA actualmente recibe mucha atención por parte de grandes empresas

IA Generativa

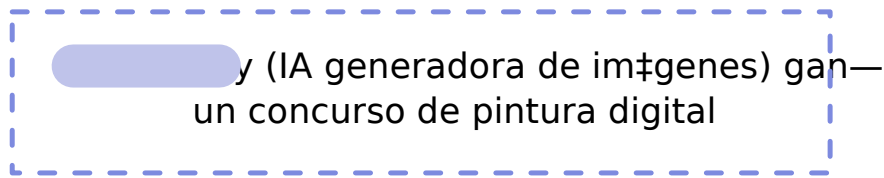
La IA Generativa permite, por ejemplo, generar la imagen de un perro a partir de la descripción de la imagen de un perro

- ¥ La idea es entrenar con imágenes y sus descripciones, por ejemplo
- ¥ Posible gracias a la alta disponibilidad de datos y una inmensa potencia

¥ Los contenidos creados mediante la IA generativa no son más que **modelos de contenidos existentes**, pero son en sí mismos únicos

IA Creativa

En 2022 la IA se volvió creativa, se crearon generadores de imágenes, de videos y nuevos agentes conversacionales



IA Fiable (Trustworthy AI)

La IA debe ser íntima, ética y robusta. Hay siete requisitos que debe cumplir si o si un sistema inteligente:

- I. Intervención y supervisión humanas.** los sistemas de IA tienen que permitir ser gobernados o supervisados por humanos.
- II. Solidez y seguridad técnicas.** Los sistemas de IA tienen que garantizar robustez tecnológica e incluso considerar planes de contingencia para la adaptación ante comportamientos anómalos.
- III. Privacidad y gestión de datos.** Los datos tienen que estar protegidos
- IV. Transparencia.** El comportamiento de los sistemas de IA debe poder ser monitorizado o trazado
- V. Diversidad, no discriminación y equidad.** El proceso de adquisición y anotación de los datos tiene que preservar la igualdad y evitar la discriminación de los ciudadanos.
- VI. Bienestar social y medioambiental.**
- VII. Rendición de cuentas.** Esta directriz está relacionada con el principio de responsabilidad.

Ética de la IA

La inteligencia artificial está cada vez más presente, y es capaz de transformar nuestras sociedades y desafía lo que significa ser humano.

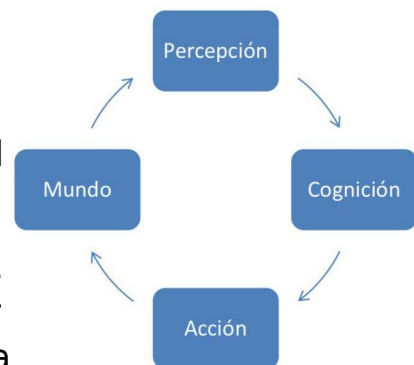
Es importante que se creen políticas reguladoras y que el enfoque de la IA siempre sea estar al servicio de las necesidades de los humanos, no al contrario

Agentes Inteligentes

La IA se inspira en la Ciencia cognitiva, la Psicología Cognitiva y la Neurociencia

Es un subcampo de la Informática dedicado a la construcción de agentes que exhiben aspectos del comportamiento inteligente

Un **Agente inteligente** es un sistema de ordenador, situado en algún entorno, que es capaz de realizar acciones de forma autónoma y que es flexible para lograr los objetivos planteados.



- ✧ **Situación:** el agente recibe entradas sensoriales de un entorno en donde está situado y realiza acciones que cambian dicho entorno
- ✧ **Autonomía:** el sistema es capaz de actuar sin la intervención directa de los humanos y tiene control sobre sus propias acciones y estado interno
- ✧ **Reactivo:** el agente debe percibir el entorno y responder de una forma temporal a los cambios que ocurren en dicho entorno
- ✧ **Pro-activo:** los agentes no deben simplemente actuar en respuesta a su entorno, deben de ser capaces de exhibir comportamientos dirigidos a lograr objetivos que sean oportunos, y tomar la iniciativa cuando sea apropiado
- ✧ **Social:** los agentes deben de ser capaces de interactuar, cuando sea apropiado, con otros agentes artificiales o humanos para completar su propio proceso de resolución del problema y ayudar a otros con sus actividades

Características de los entornos que determinan el agente apropiado a utilizar:

- ✧ **Completamente Observable:** disponemos de sensores que detectan toda la información relevante en un estado para tomar una decisión (Puzles, Juegos)
- ✧ **Parcialmente observable:** disponemos de información parcial (planificador de rutas sin información de carreteras cortadas)
- ✧ **Determinista:** el estado siguiente a la ejecución de una acción podemos determinarlo siempre (cubo de rubik)
- ✧ **No Determinista:** no podemos predecir con total certeza lo que puede ocurrir después de ejecutar una acción
- ✧ **Estático:** podemos tener todo el tiempo que queramos para encontrar solución
- ✧ **Dinámico:** tenemos que tomar una decisión de actuación rápido (pacman)

Esto no son apuntes pero **tiene un 10 asegurado** (y lo vas a disfrutar igual).

Abre la Cuenta NoCuenta con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Me interesa



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en [ing.es](https://www.ing.es)

- ¥ **Discreto**: conjunto finito de estados, acciones en intervalos discretos de tiempo
- ¥ **Continuo**: estados continuos (velocidad, posición) y acciones continuas (ángulo de giro, velocidad de giro)
- ¥ **Conocido**: conocemos todos los aspectos del mundo y su dinámica
- ¥ **Parcialmente Desconocido**: desconocemos todos los resultados de las acciones hay que explorar y aprender

Los peores entornos para una IA son: parcialmente observable, no determinista, dinámico, continuo y desconocido (tal como el mundo real), y las mejores son: completamente observables, deterministas, estáticos, discretos, conocidos

Un **Sistema Basado en Agentes** será un sistema en el que la abstracción clave utilizada es precisamente la de agente. Un **sistema multi-agente** es un sistema diseñado e implementado con varios agentes interactuando, son interesantes para representar problemas que tienen:

- Múltiples formas de ser resueltos
- Múltiples perspectivas
- Múltiples entidades para resolver el problema

La **cooperación** (trabajar juntos para resolver algo), **coordinación** (organizar una actividad para evitar las interacciones perjudiciales y explotar las beneficiosas) y **negociación** (llegar a un acuerdo que sea aceptable por todas las partes implicadas) son clave para la interacción entre agentes

SMA

una red más o menos unida de resolutores de problemas (agentes) que trabajan conjuntamente para resolver problemas que están más allá de las capacidades individuales o del conocimiento de cada resolutor del problema

- ¥ Cada agente tiene información incompleta, o no todas las capacidades para resolver el problema, así cada agente tiene un punto de vista limitado
- ¥ No hay un sistema de control global
- ¥ Los datos no están centralizados
- ¥ La computación es asíncrona

Arquitecturas de Agentes

- Arquitecturas Deliberativas

Un **agente deliberativo** es aquel que contiene un modelo simbólico del mundo

explícitamente representado, y cuyas decisiones se realizan a través de un razonamiento lógico basado en emparejamientos de patrones y manipulaciones simbólicas

google maps!

Problemas:

- ¥ Trasladar en un tiempo razonable para que sea útil el mundo real en una descripción simbólica precisa y adecuada
- ¥ Representar simbólicamente la información acerca de entidades y procesos complejos del mundo real, y como conseguir que los agentes razonen con esta información para que los resultados sean útiles

Sistema de símbolos físicos: un conjunto de entidades físicas (símbolos) que pueden combinarse para formar estructuras, y que es capaz de ejecutar procesos que operan con dichos símbolos de acuerdo a conjuntos de instrucciones codificadas simbólicamente.

La hipótesis de sistema de símbolos físicos dice que tales sistemas son capaces de generar acciones inteligentes

Arquitecturas Reactivas

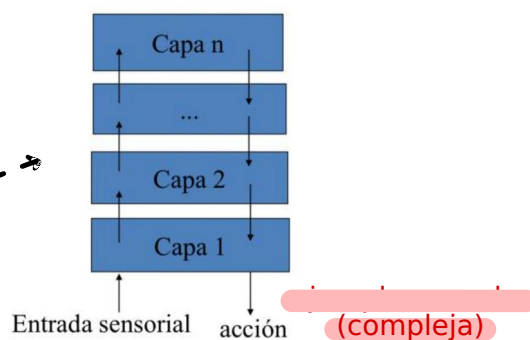
Una **arquitectura reactiva** es aquella que no incluye ninguna clase de modelo centralizado de representación simbólica del mundo, y no hace uso de razonamiento complejo.

- ¥ El comportamiento inteligente puede ser generado sin una representación explícita ni un razonamiento abstracto explícito de la clase que la IA simbólica propone.
- ¥ La inteligencia es una propiedad emergente de ciertos sistemas complejos.
- ¥ El comportamiento inteligente surge como el **resultado de la interacción del agente con su entorno**

(la + simple) !

Arquitecturas Híbridas

Las arquitecturas híbridas presentan una **estructura vertical**



(compleja)

Agentes Reactivos

Existen dos tipos de representaciones, los modelos icónicos y los modelos basados en características.

buscar + info de **de**ly

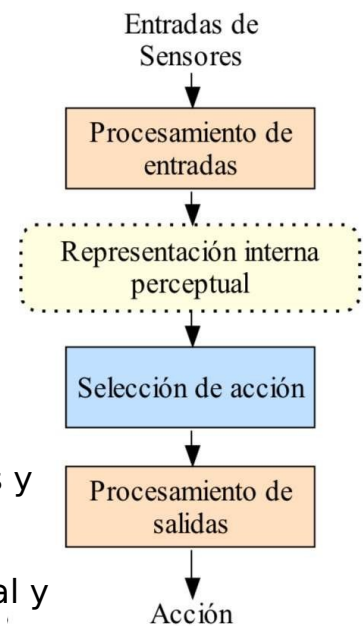
WUOLAH

Encuentra tus prácticas de empresa o tu primer trabajo temporal de lo que has estudiado en InfoJobs

El agente reactivo: sigue el esquema de percepción-acción al igual que los humanos:

1. El agente reactivo percibe su entorno a través de sensores.
2. Procesa la información percibida y hace una representación interna de la misma.
3. Escoge una acción, entre las posibles, considerando la información percibida.
4. Transforma la acción en señales para los actuadores y la realiza.

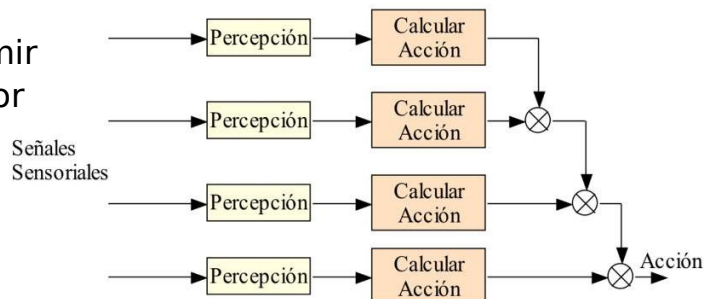
Es un proceso que lleva primero el procesamiento perceptual y segundo la fase de cálculo de la acción.



Arquitecturas de Agentes Reactivos

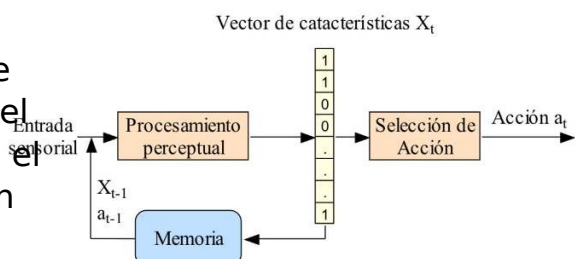
- ¥ **Sistemas de Producción:**
- ¥ **Redes Neuronales:** red de unidades lógicas con umbral
- ¥ **Arquitecturas de Subsumción:** consiste en agrupar módulos de comportamiento, cada módulo de comportamiento tiene una acción asociada, recibe la percepción directamente y comprueba una condición. Si esta se cumple, el módulo devuelve la acción a realizar.

Un módulo se puede subsumir en otro. Si el módulo superior del esquema se cumple, se ejecuta este en lugar de los módulos inferiores.



El agente reactivo con memoria: los sistemas con memoria mejoran la precisión teniendo en cuenta el historial sensorial previo.

La representación de un estado en el instante $t+1$ es función de la entradas sensoriales en el instante $t+1$, la representación del estado en el instante anterior t y la acción seleccionada en el instante anterior t



Los agentes reactivos se diseñan completamente y por tanto es necesario anticipar todas las posibles reacciones para todas las situaciones, realizan pocos cálculos y se almacena todo en memoria.

Esto no son apuntes pero **tiene un 10 asegurado** (y lo vas a disfrutar igual).

Abre la Cuenta NoCuenta con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Me interesa

1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandeses con una garantía de hasta 100.000 euros por depositante. Consulta más información en [ing.es](https://www.ing.es)



Tema 3: Búsqueda en Espacios de Estados

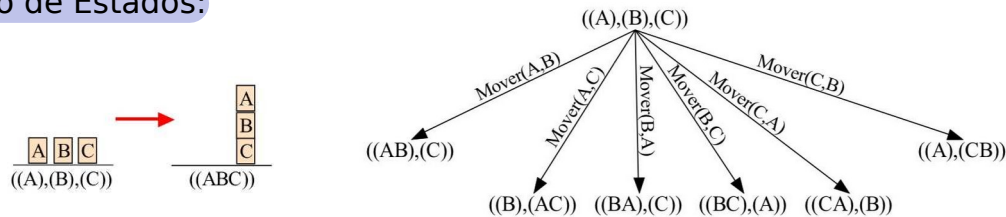
Diseño de un Agente Deliberativo

- ¥ El agente dispone de un modelo del mundo en el que habita
- ¥ El agente dispone de un modelo de los efectos de sus acciones sobre el mundo
- ¥ El agente es capaz de razonar sobre esos modelos para decidir que hacer para conseguir un objetivo

Espacio de estados: representación del conocimiento a través de las acciones del agente (un estado es una representación del mundo en un instante dado)

Búsqueda en el espacio de estados: resolución del problema mediante proyección de las distintas acciones (proyectar: imaginar que va a ocurrir después una acción)

Grafo de Estados:



A la secuencia de acciones que lleva el agente desde un estado inicial hasta un estado destino se denomina **plan**. El estudio de dicha secuencia se denomina una **planificación**

- ¥ Grafos Explícitos: construcción del grafo completo en la memoria de agente
- ¥ Grafo Implícito: el grafo se va creando y explorando a la vez sin necesidad de almacenarlo si no interesa

Un agente deliberativo resuelve un problema explorando un grafo dirigido

Factor de Ramificación
(número de acciones promedio a las que
puedo llegar desde un estado)

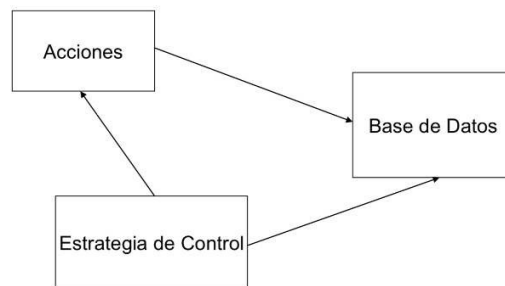
Profundidad del árbol
de Búsqueda

Ejemplo: Problema de la Aspiradora

Tenemos una aspiradora cuyo espacio de estados consta de 8 estados. La resolución del problema se puede complicar si no conoces los 8 estados de forma completa, ya que la aspiradora puede aspirar habitaciones que ya están limpias, o cuando no conoce las consecuencias de las acciones que puede realizar. En definitiva, el problema es la **INCERTIDUMBRE**.

Sistemas de Búsqueda y Estrategias

El criterio con el que funciona un agente deliberativo es la **búsqueda**. Para poder realizar esta búsqueda, es necesario estos tres elementos. Tanto las acciones como la base de datos (que almacena, entre otras cosas, el espacio de estados) son **elementos propios** del problema.



Procedimiento de Búsqueda general de la Estrategia de Control:

1. DATOS base de datos inicial
2. until DATOS satisface la condición de terminación
3. begin
4. select alguna acción A en el conjunto de acciones que pueda ser aplicada a DATOS
5. DATOS resultado de aplicar A a DATOS
6. end

Estrategias de control:

- ¥ **Estrategias irrevocables**: en cada momento, el grado explícito lo constituye un único nodo, que incluye la descripción completa del sistema en ese momento.
 1. Se selecciona una acción A
 2. Se aplica sobre el estado del sistema E, para obtener el nuevo estado $\hat{E} = A(E)$
 3. Se borra de memoria E y se sustituye por \hat{E}
- ¥ **Estrategias tentativas**: guardan información de alguna acción pasada, a diferencia de las irrevocables, que son más simples y en general más eficientes.

- **Retroactivas (Backtracking)** En memoria sólo guardamos un hijo de cada estado, se mantiene el camino desde el estado inicial hasta el actual. El grafo explícito, por tanto, es realmente un camino — **lista de estados**.


El proceso termina cuando hemos llegado al objetivo y no deseamos encontrar más soluciones, o bien no hay más operadores aplicables al nodo raíz.

El **backtracking** (o retroceso) se produce cuando:

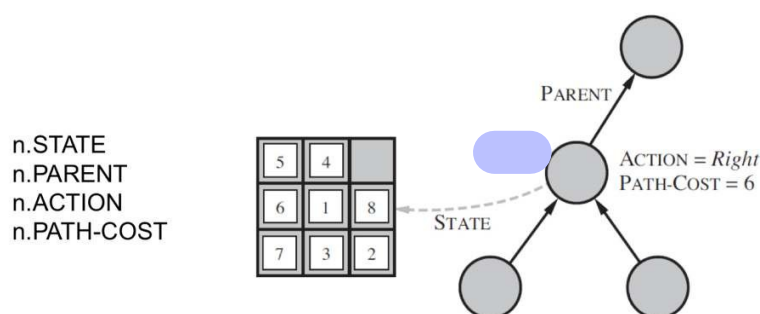
- ¥ Se ha encontrado una solución, pero deseamos encontrar otra solución alternativa.
- ¥ Se ha llegado a un límite en el nivel de profundidad explorado o el tiempo de exploración en una misma rama.
- ¥ Se ha generado un estado que ya existía en el camino **no se repite o ciclo!**
- ¥ No existen reglas aplicables al último nodo de la lista (último nodo del grafo explícito).

WUOLAH

- Búsqueda en Grafos: en memoria se guardan **todos los estados** (o nodos generados hasta el momento), de forma que la búsqueda puede proseguir por cualquiera de ellos. Esta estrategia tiene máxima complejidad y máxima flexibilidad, ya que su grafo explícito es realmente un **grafo**.

1. Seleccionar un estado E del grafo.
2. Seleccionar un operador A aplicable sobre E.
3. Aplicar A, para obtener un nuevo nodo A(E).  (exploración de grafos)
4. A-adir el arco $E \rightarrow A(E)$ al grafo
5. Repetir el proceso.

Infraestructura para los Algoritmos de Búsqueda

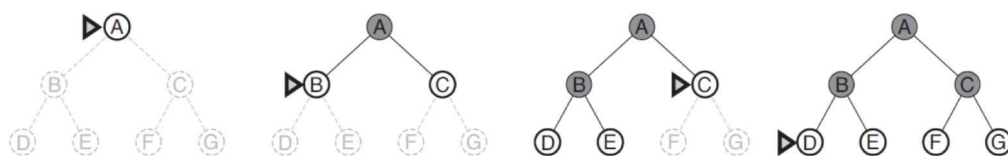


Medidas del Comportamiento de un Sistema de Búsqueda

- ¥ **Completitud**: hay garantía de encontrar la solución si esta existe
- ¥ **Optimalidad**: hay garantía de encontrar la solución óptima
- ¥ **Complejidad en tiempo**: ¿Cuánto tiempo se requiere para encontrar la solución?
- ¥ **Complejidad en espacio**: ¿Cuánta memoria se requiere para realizar la búsqueda?

Búsquedas sin Información

Búsqueda en Anchura — **Búsqueda con Costo** (exploración de grafos con una cola):



- ¥ **Completo**: encuentra la solución si existe y el factor de ramificación es finito en cada nodo
- ¥ **Optimalidad**: si todos los operadores tienen el mismo coste, encontrará la solución óptima
- ¥ **Eficiencia**: buena si las metas están cercanas
- ¥ **Problema**: consume memoria exponencial

Esto no son apuntes pero tiene un 10 asegurado (y lo vas a disfrutar igual).

Abre la Cuenta NoCuenta con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Me interesa

1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandes con una garantía de hasta 100.000 euros por depositante. Consulta más información en [ing.es](https://www.ing.es)



function BREADTH-FIRST-SEARCH(*problem*) **returns** a solution, or failure

node ← a node with STATE = *problem*.INITIAL-STATE, PATH-COST = 0

if *problem*.GOAL-TEST(*node*.STATE) **then return** SOLUTION(*node*)

frontier ← a FIFO queue with *node* as the only element

explored ← an empty set

loop do

if EMPTY?(*frontier*) **then return** failure

node ← POP(*frontier*) /* chooses the shallowest node in *frontier* */

add *node*.STATE to *explored*

for each *action* in *problem*.ACTIONS(*node*.STATE) **do**

child ← CHILD-NODE(*problem*, *node*, *action*)

if *child*.STATE is not in *explored* or *frontier* **then**

if *problem*.GOAL-TEST(*child*.STATE) **then return** SOLUTION(*child*)

frontier ← INSERT(*child*, *frontier*)

Frontier = frontera = abiertos
Explored = explorados = cerrados

Búsqueda con Costo Uniforme (exploración de grafos con cola con prioridad): misma idea que la búsqueda en anchura pero teniendo en cuenta los costos, haciendo que encuentre siempre la solución óptima

function UNIFORM-COST-SEARCH(*problem*) **returns** a solution, or failure

node ← a node with STATE = *problem*.INITIAL-STATE, PATH-COST = 0

frontier ← a priority queue ordered by PATH-COST, with *node* as the only element

explored ← an empty set

loop do

if EMPTY?(*frontier*) **then return** failure

node ← POP(*frontier*) /* chooses the lowest-cost node in *frontier* */

if *problem*.GOAL-TEST(*node*.STATE) **then return** SOLUTION(*node*)

add *node*.STATE to *explored*

for each *action* in *problem*.ACTIONS(*node*.STATE) **do**

child ← CHILD-NODE(*problem*, *node*, *action*)

if *child*.STATE is not in *explored* or *frontier* **then**

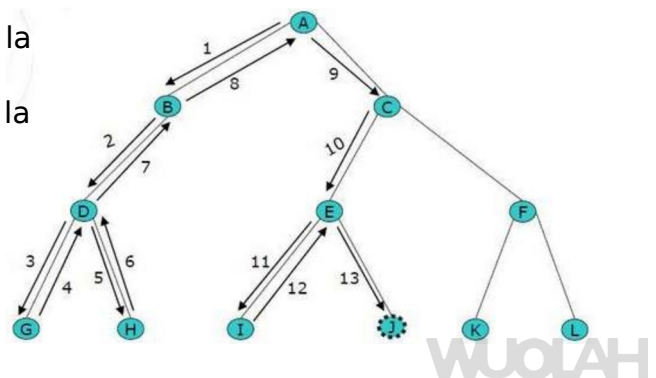
frontier ← INSERT(*child*, *frontier*)

else if *child*.STATE is in *frontier* with higher PATH-COST **then**

replace that *frontier* node with *child*

Búsqueda en Profundidad Retroactiva (exploración de grafos con una pila): Igual que la búsqueda en anchura cambiando FIFO por LIFO

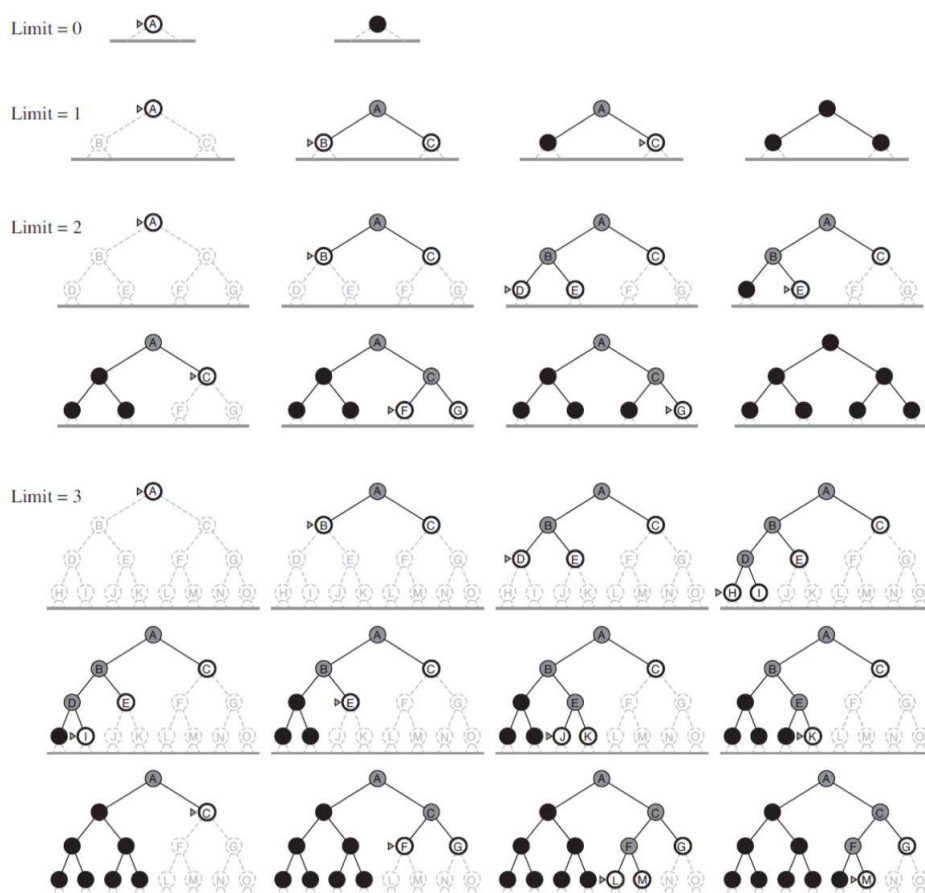
- ¥ Complejidad: no asegura encontrar la solución
- ¥ Optimalidad: no asegura encontrar la solución óptima
- ¥ Eficiencia: bueno cuando las metas están alejadas del estado inicial, o hay problemas de memoria
- ¥ No es bueno cuando hay ciclos



function DEPTH-LIMITED-SEARCH(*problem*, *limit*) **returns** a solution, or failure/cutoff
return RECURSIVE-DLS(MAKE-NODE(*problem*.INITIAL-STATE), *problem*, *limit*)

function RECURSIVE-DLS(*node*, *problem*, *limit*) **returns** a solution, or failure/cutoff
if *problem*.GOAL-TEST(*node*.STATE) **then** **return** SOLUTION(*node*)
else if *limit* = 0 **then** **return** cutoff
else
 cutoff_occurred? \leftarrow false
 for each *action* **in** *problem*.ACTIONS(*node*.STATE) **do**
 child \leftarrow CHILD-NODE(*problem*, *node*, *action*)
 result \leftarrow RECURSIVE-DLS(*child*, *problem*, *limit* - 1)
 if *result* = cutoff **then** *cutoff_occurred?* \leftarrow true
 else if *result* \neq failure **then** **return** *result*
 if *cutoff_occurred?* **then** **return** cutoff **else** **return** failure

Descenso Iterativo: es una técnica retroactiva que intenta simular la búsqueda en anchura.



function ITERATIVE-DEEPENING-SEARCH(*problem*) **returns** a solution, or failure
for *depth* = 0 **to** ∞ **do**
 result \leftarrow DEPTH-LIMITED-SEARCH(*problem*, *depth*)
 if *result* \neq cutoff **then** **return** *result*

En general tiene las características de la búsqueda en anchura, es más eficiente en memoria, pero los nodos se regeneran, es decir, se exploran varias veces los mismos nodos. Genera más nodos que la búsqueda en anchura, pero no significa que sea necesariamente más lento ya que depende de otras cosas que no se pueden controlar.

WUOLAH

Búsqueda Bidireccional: realizar dos búsquedas por ambos lados. Una de las ventajas es que reduce el tiempo de búsqueda, por ejemplo en un problema con factor de ramificación 20, se reparten 10 nodos en cada dirección.

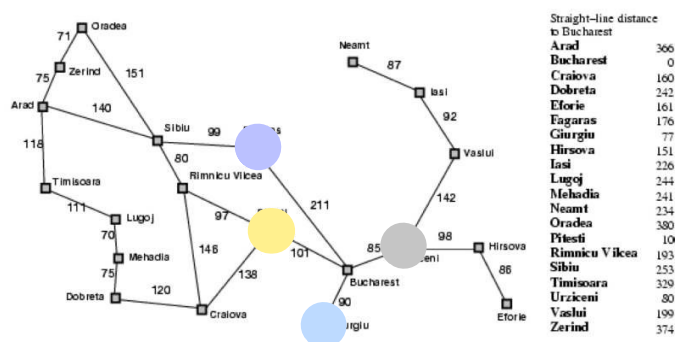
Los métodos estudiados hasta ahora difícilmente resuelven problemas complejos, de hecho, los humanos para resolver problemas no hacen búsquedas exhaustivas, si no que toman decisiones en base a una información que tiene.

Búsquedas con Información

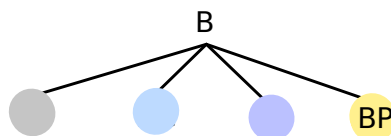
Heurísticas del griego "heurisko", yo encuentro

Las **heurísticas** son criterios, métodos o **principios para decidir** cuál de entre varias acciones promete ser la mejor para alcanzar una determinada meta. En otras palabras, es algo que te ayuda a elegir de entre varias opciones.

En IA, entendemos por heurística un método para resolver problemas que en general **no garantiza la solución óptima**, pero que en media produce resultados satisfactorios en la resolución de un problema.



La heurística de este problema combina los costos con la distancia en línea recta desde la ciudad en donde estoy hasta la ciudad destino.

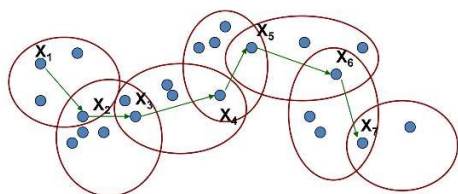


En IA, implementaremos heurísticas como **funciones que devuelven un valor numérico**, cuya maximización o minimización guiará al proceso de búsqueda a la solución.

Métodos de Escalada (estrategias irrevocables)

En función de cómo se enfoque la heurística, se pueden dar **soluciones parciales** o **completas**. Las soluciones completas suelen dar mejores resultados, estos métodos funcionan mejor con ellas, pero, aunque será más complicado, también van con parciales.

Si dibujamos las soluciones como puntos en el espacio, una búsqueda local consiste en seleccionar la solución mejor en el vecindario de una solución inicial, e ir viajando por las soluciones del espacio hasta encontrar un óptimo (local o global)



En inglés se llama **"Hill Climbing"**, ya que para llegar a la solución toma decisiones de tipo local, en base a las casillas más próximas que tiene (casillas en un mundo cuadrado). Siempre intentar mejorar la solución **a paso**, sin recordar lo anterior.

Esto no son apuntes pero tiene un 10 asegurado (y lo vas a disfrutar igual).

Abre la Cuenta NoCuenta con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Me interesa



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en [ing.es](https://www.ing.es)

Algoritmo de Escala Simple

E: Estado activo

```
while (E no sea el objetivo
y queden nodos por explorar a partir de E) {
  Seleccionar operador A para aplicarlo a E
  Evaluar f(A(E))
  if (f(A(E)) > f(E)) {
    E = R(E)
  }
}
```

Algoritmo de Escala Por Máxima Pendiente

E: Estado activo

```
while (E no sea el objetivo
y queden nodos por explorar a partir de E) {
  Para todos los operadores Ai, obtener Ei = Ai(E)
  Evaluar f(Ei) para todos los estados Ei = Ai(E)
  Seleccionar Emax tal que f(Emax) = max{f(Ei)}
  if (f(Emax) > f(E)) {
    E = Emax
  } else return E
}
```

- ✖ **Complejidad:** no tiene porque encontrar la solución
- ✖ **Admisibilidad:** no siendo completo, aun menos será admisible
- ✖ **Eficiencia:** rápido y útil si la función es monótona (de)creciente

Algunas Variaciones Estocásticas: intentan mejorar la exploración con criterios de tipo aleatorio

- ✖ Algoritmo de escalada **estocástico**: tengo un estado e , y a partir de ese, puedo llegar a otros tres estados e_1 , e_2 y e_3 , y la heurística es de minimizar. Para evitar que vaya siempre por el estado que siempre da el resultado óptimo, se crea un criterio probabilístico que está relacionado con la heurística. De esta forma casi siempre saldrá el estado que tiene la mejor valoración, pero alguna vez saldrá otro estado.
-
- ✖ Algoritmo de escalada de **primera opción**: se hace un sorteo y se elige aleatoriamente el orden de cada estado en cada paso sin tener en cuenta la función heurística
 - ✖ Algoritmo de escalada de **reinicio aleatorio**: genera muchos puntos de partida muy rápidos aleatoriamente y se aparece en uno
 - ✖ **Enfriamiento simulado**: Es un método de búsqueda local, que se basa en principios de Termodinámica. Al contrario que otros métodos de ascensión de colinas, permite visitar soluciones peores que la actual para evitar óptimos locales.

Programa de Enfriamiento: cómo defino la temperatura de inicio y cómo cambia en cada iteración, siendo la temperatura un parámetro artificial

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state

inputs: *problem*, a problem

schedule, a mapping from time to "temperature"

current ← MAKE-NODE(*problem*.INITIAL-STATE)

for $t = 1$ **to** ∞ **do**

$T \leftarrow \text{schedule}(t)$

if $T = 0$ **then return** *current*

next ← a randomly selected successor of *current*

$\Delta E \leftarrow \text{next.VALUE} - \text{current.VALUE}$

if $\Delta E > 0$ **then** *current* ← *next*

else *current* ← *next* only with probability $e^{\Delta E/T}$

Si estoy en un estado y el siguiente es mejor, cambio a él.

Si es peor, calculo p ($p = e^{\Delta E/T}$), que es la diferencia de la función heurística en el estado actual y en el estado anterior.

Consulta condiciones aquí



do your thing

Ventajas:

- ¥ Al ser un método probabilístico, tiene capacidad para salir de óptimos locales.
- ¥ Es eficiente.
- ¥ Es fácil de implementar.

Inconvenientes:

- ¥ Encontrar la temperatura inicial T_i , el método de actualización de temperatura ³, el número de vecinos a generar en cada estado y el número de iteraciones óptimo es una tarea que requiere de muchas pruebas de ensayo y error hasta que ajustamos los parámetros óptimos.

Pese a todo, el algoritmo puede proporcionar soluciones mucho mejores que utilizando algoritmos no probabilísticos.

Algoritmos Genéticos

Son algoritmos de optimización basados en el proceso de la evolución natural de Darwin.

En un proceso de **evolución**, existe una población de individuos. Los más adecuados a su entorno se reproducen y tienen descendencia (a veces con **mutaciones** que mejoran su idoneidad al entorno). Los más adecuados sobreviven para la siguiente generación.

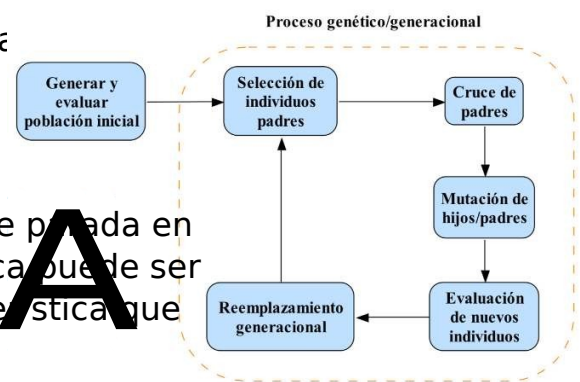
- ¥ No necesitan partir de un nodo/estado inicial: ¿Hay toda una población!
- ¥ Su objetivo es encontrar una solución cuyo valor de función objetivo sea óptimo.

Cromosoma ³ Vector representación de una solución al problema
Gen ³ Variable/Atributo concreto del vector de representación de una solución
Población ³ Conjunto de soluciones al problema (uno solo no evoluciona)
Adecuación al entorno ³ Valor de función objetivo (fitness, heurística)
Selección natural ³ Operador de selección
Reproducción sexual ³ Operador de cruce
Mutación ³ Operador de mutación
Cambio generacional ³ Operador de reemplazamiento

Tienen varias características muy específicas, usan métodos de escalada pero lo nuevo que añade es que tengo que tener una estructura que lista cuáles son los **estados** y las **operaciones**. Tengo que **adaptar mi problema** al algoritmo, el algoritmo no se adapta al problema.

En la población, hay una probabilidad dada a priori de que un individuo pueda **mutar**. A su vez, cuando un individuo muta, existe otra probabilidad de que cada gen mute o no.

Hay muchas formas de elegir la condición de parada en esta bucle, por ejemplo, la función heurística puede ser que pare cuando ya no haya ninguna característica que sea distinta y mejor en algún descendiente.



Búsqueda Primero el Mejor (exploración de grafos)

También se suelen llamar al mejor nodo, ya que hay una valoración heurística que determina cuál es el mejor de los nodos pendientes, pero manteniendo el grafo.

Son los que tienen más garantías, no solo para encontrar solución sino para encontrar una solución óptima. Por esto mismo también son los métodos más complejos tanto en implementación como en computación.

Búsqueda Primero Mejor Greedy (BFS)

Tiene una lista de nodos pendientes por los que puede seguir explorando y elige el mejor para avanzar. No siempre da la solución más óptima (el camino más corto). Es más una idea que un algoritmo.

Algoritmo A*

Su objetivo es buscar el **camino de coste mínimo** (siempre busca la solución más óptima), sigue la heurística: $f(n) = g(n) + h(n)$, donde g es el coste de n al inicio y h es la estimación del coste de n al objetivo.

Es una búsqueda en grafos donde frontier (abierto) es una cola con prioridad ordenada de acuerdo a $f(n)$.

FRONTIER contiene el nodo inicial, EXPLORED está vacío.

Comienza un ciclo que se repite hasta que se encuentra solución o hasta que frontier queda vacío.

- ¥ Seleccionar el mejor nodo de frontier
 - Si es un nodo objetivo, terminar
 - En otro caso se expande dicho nodo
- ¥ Para cada uno de los nodos sucesores
 - Si está en frontier insertarlo manteniendo la información del mejor padre
 - Si está en explored insertarlo manteniendo la información del mejor padre y actualizar la información de los descendientes
- ¥ En otro caso, insertarlo como un nodo nuevo

estructura de un nodo
(estado, hijos, mejor_padre, g & h)

Para nodo n Repetidos:

caso 1: el nuevo padre de n no es mejor que el padre anterior. FIN

caso 2: el nuevo padre de n es mejor que el padre anterior.

- Actualizar enlace al padre, actualizar el nuevo valor de coste del camino:
- Propagar la información a los hijos de n

Casos Particulares del Algoritmo A*:

¥ Supongamos que usamos el algoritmo A* pero tomamos siempre $h(n)=0$, entonces, en el caso que el coste de cada arco sea siempre unidad (1), el algoritmo se comporta como la **búsqueda en anchura**. En otro caso, el algoritmo se comporta como la búsqueda de **coste uniforme**. Si el coste de cada arco es -1 entonces será búsqueda en **profundidad**.

Esto no son apuntes pero **tiene un 10 asegurado** (y lo vas a disfrutar igual).

Abre la Cuenta NoCuenta con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Me interesa

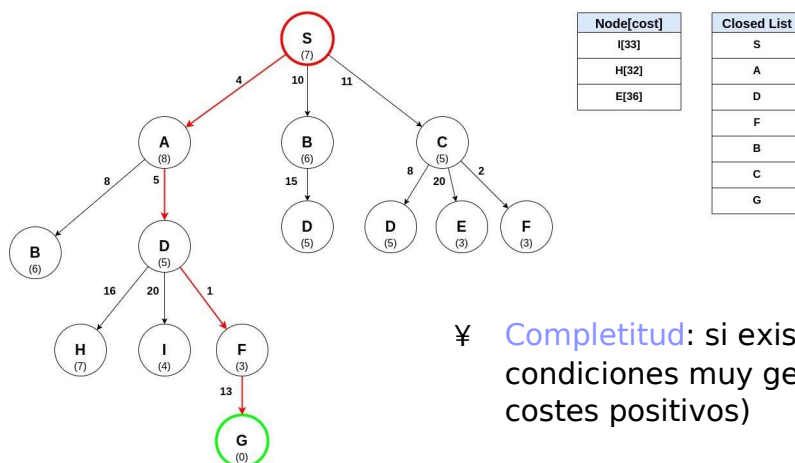
1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en [ing.es](#)



¿Supongamos que usamos el algoritmo A* pero tomamos siempre $g(n)=0$, entonces el algoritmo se comporta como el algoritmo de **búsqueda primero el mejor greedy**.



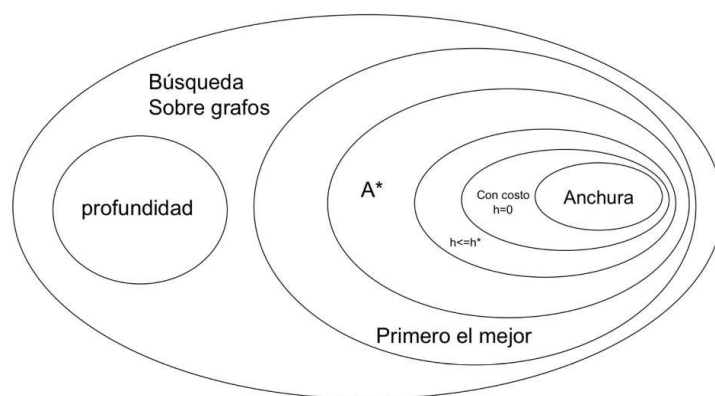
¿ **Complejidad**: si existe solución, la encuentra bajo condiciones muy generales (sucesores finitos y costes positivos)

¿ **Admisibilidad**: si hay una solución óptima, bajo unas condiciones muy generales y si la función $h(n)$ es admisible: $h(n) \leq h^*(n)$

Búsqueda Dirigida (Beam Search)

Una variación del algoritmo A* que **limita el factor de ramificación** en problemas complejos

- ¿ Cada vez que se expande un nodo, se generan sus sucesores, se evalúan con la función heurística f , y se eliminan aquellos sucesores con peor valor de la f , quedándonos con un número fijo de sucesores. En otras palabras, se realiza una poda a priori y se mantienen los k mejores



Problemas y Dificultades

- ¿ Los procesos de percepción no siempre pueden obtener la información necesaria acerca del estado del entorno
- ¿ Las acciones pueden no disponer siempre de modelos de sus efectos
- ¿ Pueden haber otros procesos físicos, u otros agentes, en el mundo
- ¿ En el tiempo que transcurre desde la construcción de un plan, el mundo puede cambiar de tal manera que el plan ya no sea adecuado

WUOLAH

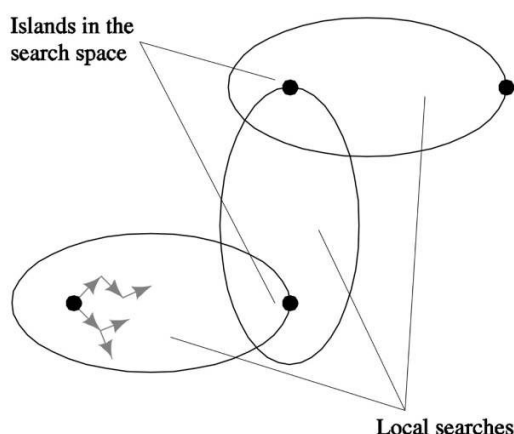
- ¥ Podr'a suceder que se le requiriese al agente actuar antes de que pudiese completar una bœsqueda de un estado objetivo
- ¥ Aunque el agente dispusiera de tiempo suficiente, sus recursos de memoria podr'an no permitirle realizar la bœsqueda de un estado objetivo

Problemas Descomponibles y Bœsqueda

Heur'sticas sobre el Proceso de Bœsqueda

Las heurísticas pueden venir en la propia definición del problema o en su traza, no tienen por quē aplicarse solo en la función heurística.

¥ Bœsqueda Orientada a Subobjetivos



Una de las posibilidades de aplicar heur'stica, es que, para poder llegar al objetivo, hay que pasar obligatoriamente por los **subobjetivos**. Dividiendo as' el proceso de bœsqueda y a-adiendo dichos subobjetivos en la heurística

No se puede aplicar siempre, depende de la capacidad de **divisi—n del problema**. Por ejemplo, para el 8 puzzle no parece posible

¥ Bœsqueda con Horizonte

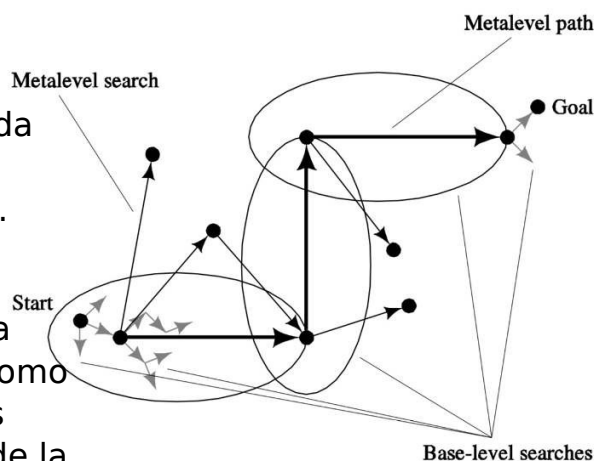
Se usa cuando se sabe que el espacio de bœsqueda es tan grande que se que no puedo explorarlo completamente. Se establece una **profundidad m'xima** (horizonte) y se realiza la bœsqueda con esa profundidad m'xima. A veces, es necesario cambiar el criterio de bœsqueda del objetivo

¥ Bœsqueda Jer'rquica

Se basan en conocimiento jer'rquico.

Por ejemplo, si estoy en la mesa y quiero salir de la habitaci—n, este tipo de bœsqueda primero se queda en el **nivel m's alto** y traza la soluci—n (ir de la mesa a la puerta).

Despu'zs, se basa en esta soluci—n y crea subsoluciones (flechas grises) basadas en la **descomposici—n heursitica del problema**, como puede ser levantarse de la silla, ponerse las zapatillas, apagar la luz, y finalmente salir de la habitaci—n.



a	6
6	4
b	0
0	4
f	6
f	9
f	3
3	5
5	9
9	5
e	8
f	4
a	8
b	8
8	8
7	7
a	8
8	9
8	8
b	8
d	5
5	0
0	0
b	0

- 4b046



c

If

- 35

5



5



14

14



- | | | | | | | | |
|---|---|---|---|---|---|---|---|
| a | b | 8 | 8 | 7 | a | 8 | 9 |
|---|---|---|---|---|---|---|---|

b
d

Esto no son apuntes pero **tiene un 10 asegurado** (y lo vas a disfrutar igual).

Abre la Cuenta NoCuenta con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Me interesa

1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en [ing.es](https://www.ing.es)



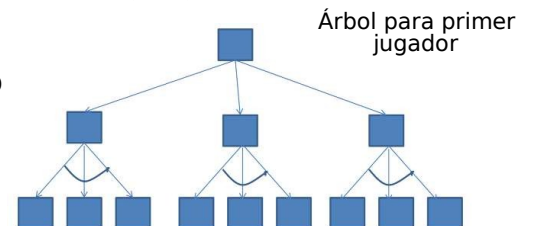
Tema 4: Búsqueda con Adversarios, Juegos

Juegos Bipersonales con Información Perfecta

Un juego de información perfecta es aquel en los jugadores tienen a su disposición toda la información de la situación del juego.

Un árbol del juego es una representación explícita de todas las formas de jugar a un juego

- Correspondencia entre árboles de juegos y árboles Y/O



Árboles de Exploración de Juegos

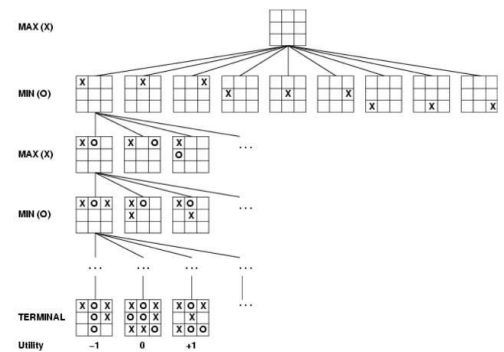
Notación MIN-MAX

■: primer jugador

■: segundo jugador

- Nodos MAX y nodos MIN

- Los nodos terminales se etiquetan con V, D o E desde el punto de vista de MAX



Algoritmo STATUS

Si J es un nodo **MAX no terminal**, entonces STATUS(J)=

- V si alguno de los sucesores de J tiene STATUS V
- D si todos los sucesores de J tienen STATUS D
- E en otro caso

Si J es un nodo **MIN no terminal**, entonces STATUS(J)=

- V si todos los sucesores de J tienen STATUS V
- D si alguno de los sucesores de J tiene STATUS D
- E en otro caso

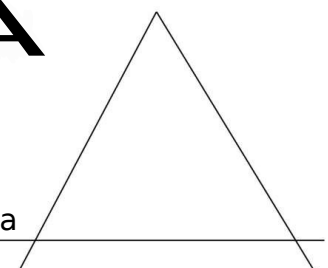
Es un algoritmo que resuelve un problema de forma eficiente, —sea que no entra dentro de la definición de IA. Los juegos complejos no se pueden resolver ya que es imposible la **exploración total** hasta la terminación. Por eso nace un **nuevo objetivo**: encontrar una buena jugada inmediata. Es importante de la heurística en el proceso.

El Modelo Básico

Arquitectura Percepción/Planificación/Actuación

Este tipo de arquitecturas analiza el estado actual, planifica una acción y la realiza, así sucesivamente.

El horizonte es la **profundidad** hasta la que buscamos, la heurística estima la **valoración del status** (v, d, e). La complejidad de un juego es B^p , donde B es el factor de ramificación y P la profundidad, es decir, el número de llamadas a la función heurística.



WUOLAH

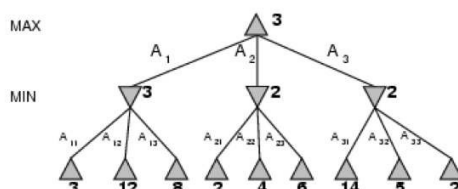
La Regla MINIMAX

El valor $V(J)$ de un nodo J de la frontera de búsqueda es igual al de su evaluación estática; en otro caso

- ¥ Si J es un **nodo MAX**, entonces su valor $V(J)$ es igual al **máximo** de los valores de sus nodos sucesores
- ¥ Si J es un **nodo MIN**, entonces su valor $V(J)$ es igual al **mínimo** de los valores de sus nodos sucesores.

Para determinar el valor minimax, $V(J)$ de un nodo J , hacer lo siguiente:

- ¥ Si J es un nodo terminal, devolver $V(J)=f(J)$; en otro caso
 - Para $k=1, 2, \dots, b$, hacer:
 - Generar J_k , el k -ésimo sucesor de J
 - Calcular $V(J_k)$
 - Si $k=1$, hacer $AV(J) \leftarrow V(J_1)$; en otro caso, para $k \geq 2$, hacer $AV(J) \leftarrow \max\{AV(J), V(J_k)\}$ si J es un nodo MAX o hacer $AV(J) \leftarrow \min\{AV(J), V(J_k)\}$ si J es un nodo MIN
- ¥ Devolver $V(J)=AV(J)$



La heurística en tableros suele ser: infinito si el tablero es de victoria, menos infinito si es un tablero de derrota y para los demás valores se queda en la valoración

Poda ALFA-BETA

Es una **mejora del algoritmo minimax**, siempre da el valor minimal. La poda se realiza de forma que siempre se podan los valores que no están entre la cota alfa y la cota beta.

	Para nodos	Se calcula	es
Cota alfa desde el nodo actual a la raíz	Nodos MIN	Máximo de los nodos MAX	Cota inferior
Cota beta	Nodos MAX	Mínimo de los nodos MIN	Cota superior

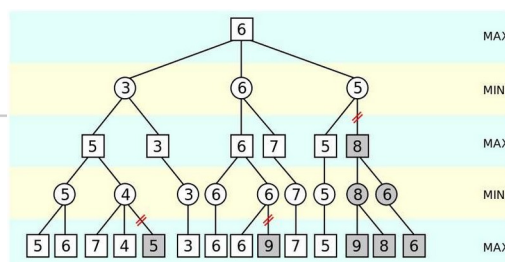
Para calcular el valor $V(J, \alpha, \beta)$:

```
función alfa-beta(nodo //en nuestro caso el tablero, profundidad,  $\alpha$ ,  $\beta$ , jugador)
  si nodo es un nodo terminal o profundidad = 0
    devolver el valor heurístico del nodo
  si jugador1
    para cada hijo de nodo
       $\alpha := \max(\alpha, \text{alfa-beta}(\text{hijo}, \text{profundidad}-1, \alpha, \beta, \text{jugador2}))$ 
      si  $\beta \leq \alpha$ 
        romper (* poda  $\beta$  *)
    devolver  $\alpha$ 
  si no
    para cada hijo de nodo
       $\beta := \min(\beta, \text{alfa-beta}(\text{hijo}, \text{profundidad}-1, \alpha, \beta, \text{jugador1}))$ 
      si  $\beta \leq \alpha$ 
        romper (* poda  $\alpha$  *)
    devolver  $\beta$ 
```

(* Llamada inicial *)
 $\text{alfa-beta}(\text{origen}, \text{profundidad}, -\text{infinito}, +\text{infinito}, \text{jugador_deseado})$

La complejidad en **espacio** es B^P

En el **tiempo**,
si es par $2B^{P/2}-1$,
si es impar $B^{(P+1)/2}+B^{(P-1)/2}-1$



Heurísticas para la Búsqueda en árboles de Juego

- ✖ **Evaluación hacia atrás:** análisis de la regla minimax
 - ✖ **Profundidad de la búsqueda:** cuanto más profunda, mejor será la solución, pero requiere mucha computación (solución: técnica de posición de reposo)
 - ✖ **Ordenación de la búsqueda:** algoritmo de la ordenación fijada, tiene más posibilidades de poda que el alfa beta, algoritmo de ordenación dinámica, reordena los nodos que, aunque tiene mucha capacidad de poda, no supera al algoritmo alfa-beta porque hace falta almacenar demasiados nodos y deja de ser una implementación retroactiva
 - ✖ **Anchura de la búsqueda**
 - ✖ **Alternativas de la búsqueda**
-

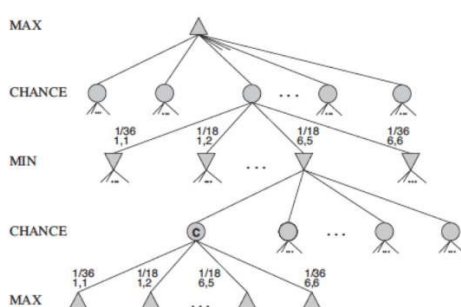
Las principales **debilidades** de la poda alfa- beta son:

- ✖ La complejidad cuando el factor de ramificación crece.
- ✖ La definición de una buena función heurística

Se ha hecho una evolución de los juegos hacia una estrategia llamada **MCTS**, Búsqueda en árboles Monte Carlo (Monte Carlo Tree Search):

- ✖ No hace uso de funciones heurísticas, sino que estima el **valor de un estado** como la **utilidad promedio** sobre un número de **simulaciones de juegos completos** empezando en un estado concreto. Hay que determinar qué jugada hace cada jugador durante la simulación (aleatoria o mediante políticas), y se sigue un patrón de **exploración/explotación**

Juegos en los que Interviene un Elemento Aleatorio



Dado que el avance depende de un elemento aleatorio (p ej un dado), la regla de propagación que se sigue es el promedio o **esperanza matemática**.

Esta aleatoriedad hace que el **factor de ramificación** sea altísimo. En los humanos sucede algo parecido, en juegos que dependen de un elemento aleatorio, no se puede planificar las acciones ni garantizar la victoria por muy bien que se juegue.

Para este tipo de juegos, es más difícil profundizar y es más difícil establecer la heurística.

Esto no son apuntes pero **tiene un 10 asegurado** (y lo vas a disfrutar igual).

Abre la Cuenta NoCuenta con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Me interesa



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en [ing.es](#)

Tema 6: Introducción al Aprendizaje

Automático

El aprendizaje es una **capacidad fundamental** de la inteligencia humana, que nos permite adaptarnos a cambios de nuestro entorno, desarrollar una gran variedad de habilidades y adquirir experiencia en nuevos dominios.

El **aprendizaje automático** son, en definitiva, programas que mejoran su comportamiento con la experiencia, y cubre una amplia gama de fenómenos como el perfeccionamiento de la habilidad y la adquisición de conocimiento.

Los programas de IA primitivos tienen el **límite** en el conocimiento que se les ha dado y no resuelven problemas más allá de esos límites, mientras que este tipo de aprendizaje **modifica el mecanismo de decisión** del agente para mejorar su comportamiento.

Los usos principales del aprendizaje automático son varios:

- ¥ Tareas difíciles de programar (reconocimiento de caras, voz, ...)
- ¥ Aplicaciones auto adaptables (interfaces inteligentes, spam killers, sistemas recomendadores, ...)
- ¥ Minería de datos (análisis de datos inteligente)

Un programa de ordenador se dice que aprende de la experiencia E con respecto a alguna clase de tareas T y a alguna medida de comportamiento P , si su comportamiento en tareas de T , medido a través de P , mejora con la experiencia E .

Estrategias de Aprendizaje

- ¥ Aprendizaje memorístico
 - ¥ Aprendizaje a través de consejos
 - ¥ Aprendizaje en la resolución de problemas
 - ¥ Aprendizaje por refuerzo (bicicleta) → el objetivo es aprender la función f un ejemplo es un par $(x, f(x))$
 - ¥ Aprendizaje basado en explicaciones
 - ¥ Aprendizaje a través de descubrimiento
 - ¥ Aprendizaje por analogía
- problema encontrar una hipótesis h tal que $h=f$ sobre los conjuntos de ejemplos de entrenamiento

Uno de los puntos clave para el aprendizaje es el tipo de **realimentación** disponible en el proceso:

- ¥ **Aprendizaje supervisado**: aprender una función a partir de ejemplos de sus entradas y salidas.
- ¥ **Aprendizaje no supervisado**: Aprender a partir de patrones de entradas para los que no se especifican los valores de sus salidas, sirve para saber la distribución de los datos o si siguen alguna tendencia

