



Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

Unidad 1 Introducción y resolución de problemas

11/01/2014





Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

Unidad 1 Lección 1

Introducción a la Inteligencia Artificial



Contenidos

1. ¿Qué es inteligencia artificial?
2. Perspectiva histórica
3. Tipos de sistemas IA
4. Marco I+D+I (Nacional y Europeo)

Objetivos

- Obtener una vision amplia del concepto Inteligencia Artificial
- Conocer el marco histórico de la Inteligencia Artificial
- Taxonomia y tipos de sistemas dentro de la Inteligencia Artificial

¿Qué es Inteligencia Artificial?

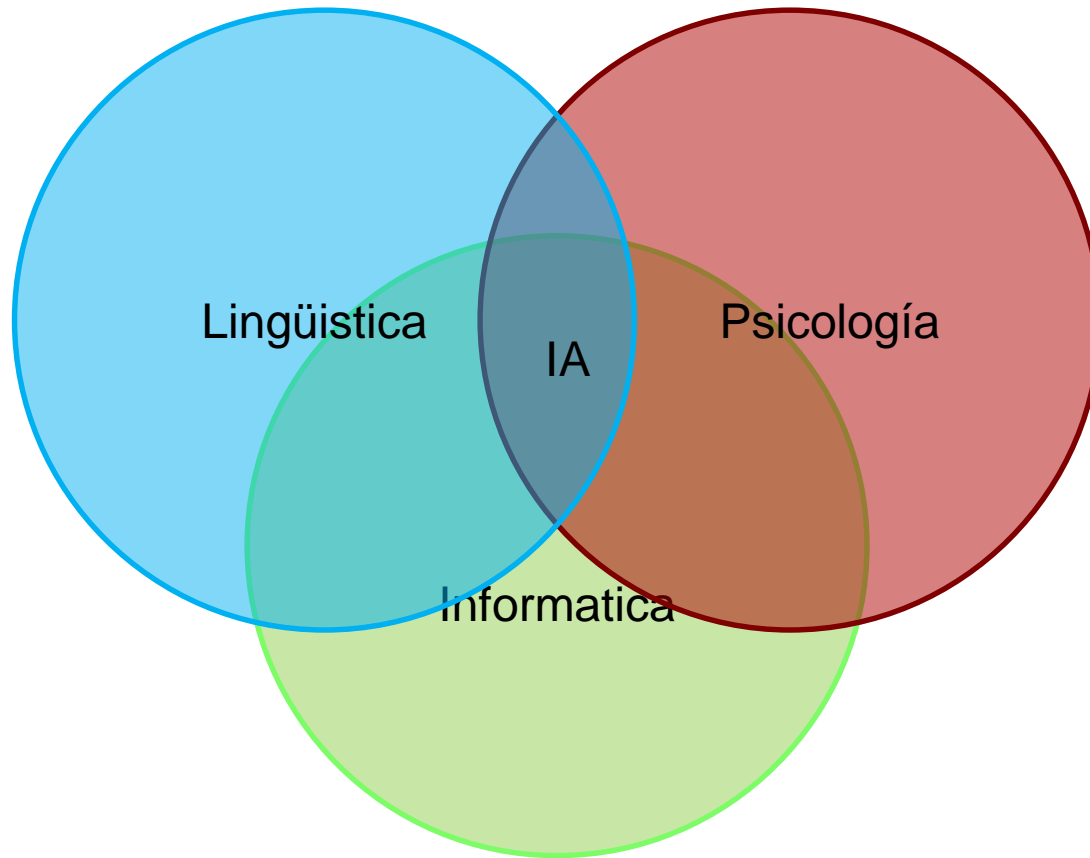
“El estudio de las facultades mentales mediante el uso de modelos computacionales” (Charniak y McDermott, 1985).

“IA (...) está relacionada con conductas inteligentes en artefactos” (Winston, 1992).

“El estudio de cómo lograr que los computadores realicen tareas que, por el momento, los humanos hacen mejor” (Rich y Knight, 1991).

“Capacidad de entender, asimilar, elaborar información y utilizarla adecuadamente.”

¿Qué es Inteligencia Artificial?



¿Qué es Inteligencia Artificial?

- Tipos de sistemas en IA:
 - Sistemas basados en conocimiento
 - Definición de dicho conocimiento a través de un experto
 - Sistemas basados en el aprendizaje
 - El conocimiento puede ser aprendido a partir de casos concretos y ejemplos
- Son tratados de manera diferente y las técnicas utilizadas también difieren pero se pueden complementar el uno al otro

¿Qué es Inteligencia Artificial?

- Dimensiones de la investigación de IA:
 - Desarrollo de nuevas funcionalidades: se centra en resolver problemas a través del uso de la computación que hasta ahora o de otro modo no pueden ser resueltos (p.e. OCR, reconocimiento de objetos, etc.).
 - Métodos y herramientas utilizadas en los sistemas: entornos de desarrollo de sistemas expertos (p.e. CLISP, Prolog).
 - Desarrollo y utilización en aplicaciones comerciales: ciclos de 5-5-5 (investigación-desarrollo-diseminación).

Perspectiva histórica



- **Fundamentos (400 a.c.)**

- Aristóteles (384-322 a.c.) Entendimiento a través de la razón.
- Formalización de los algoritmos de al-Khowarazmi (s. IX)
- primeras máquinas de cálculo de Pascal (s. XVII) y Charles Babagge (s.XIX)

- **Primeros computadores (sobre 1940)**

- Z-3 fue inventado por Konrad Zuse en 1941. En el Reino Unido, el primer sistema de Alan Turing en 1940, y el Colossus en 1943. En estados Unidos, el ABC entre 1940 y 1942 por John Atanasoff

- **ENIAC** 1946(Electronic Numerical Integrator and Computer).

Perspectiva histórica

- **Génesis (1943-1956)**

- McCulloch y Pitts han sido reconocidos como los autores del primer trabajo de IA, en 1943, proponiendo un modelo constituido por neuronas artificiales
- Primeros programas de juego del ajedrez desarrollados por Shannon y Turing entre 1950
- primeros sistemas de traducción automática, como el experimento Georgetown-IBM

Seminario de Darmouth 1956 origen del término inteligencia artificial



Biblioteca Bake, Universidad de Darmouth.

Perspectiva histórica

- Entusiasmo inicial, grandes esperanzas (1952-1969)
 - Gran éxito tanto a nivel de universidad como de empresas y centros de investigación
 - Desarrollo de herramientas como LISP
- Crisis / redimensionamiento de problemas (1966-1973)
 - Limitaciones computacionales y de hardware no son el único problema
 - Decremento de la financiación y se interrumpen un gran número de proyectos

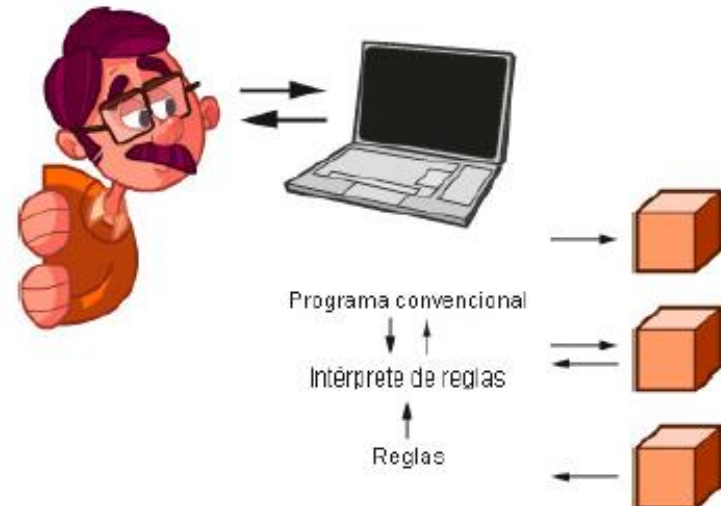
Perspectiva histórica

- Resurgimiento centrado en sistemas basados en el conocimiento (1969-1979)
 - Surge un renovado interés por sistemas expertos y basados en el conocimiento aplicado a dominios como diagnóstico médico o control de plantas
- La industria de la IA (1980 hasta el presente)
 - Sistemas expertos comienzan a reportar beneficios en sus diversas aplicaciones (p.e. DEC sistemas de pedidos)
 - Comienzan a desarrollarse nuevos aspectos de IA como la minería de datos o tecnología semántica
- 2013-....
 - Sistemas de análisis y aprendizaje en tiempo real y ubicuos para adaptarse a las variabilidades y necesidades del entorno (Big Data).

Taxonomía de la IA

- Sistemas basados en conocimiento
 - Dispone de una base de conocimiento
 - Necesidad de adquisición, formalización, y codificación
 - Los sistemas expertos pertenecen a esta categoría

Un sistema experto es un sistema computerizado que utiliza conocimiento de un dominio para resolver un problema específico de ese dominio, de forma que la solución debe ser esencialmente la misma que la proporcionada por un experto humano en ese dominio.



Taxonomía de la IA

- **Sistemas basados en aprendizaje**
 - Se compone de una fase de entrenamiento y una fase operativa
 - La fase de entrenamiento utiliza ejemplos para crear un modelo.
 - La fase operativa ejecuta el modelo aprendido para tomar decisiones.
 - OCR o sistemas de reconocimiento de objetos en imagen son dos ejemplos de este tipo de sistemas.

Taxonomía de la IA

- Redes neuronales, las Redes bayesianas, Algoritmos genéticos y Razonamiento basado en casos



Aplicaciones comerciales y sistemas experimentales

- Sistemas de control en transporte ferroviario
- Sistemas de diagnóstico y reparación (medicina y automoción)
- Agentes inteligentes en videojuegos y comercio electrónico
- Minería de datos es común en astronomía, biología, teledetección, análisis contenidos Web
- Detección de fraude a través del análisis de patrones

Aplicaciones comerciales y sistemas experimentales

- Previo a la comercialización se produce un ciclo de 5-5-5 (investigación-desarrollo-comercialización)
- Algunos centros de referencia son:
 - MIT(Instituto de Tecnología de Massachusetts)
 - KSL de Standford
 - PARC de Xerox
 - ATT Labs
 - IBM Watson Laboratory
 - Instituto de Investigación en Inteligencia Artificial del Consejo Superior de Investigaciones Científicas
- Muchas universidades también disponen de un grupo de IA (en la Universidad Europea se creó en 1999 el grupo de investigación en Sistemas Inteligentes)

Resumen

- Enfocado en esta asignatura como Informática Aplicada Avanzada y disciplina multidisciplinar
- Perspectiva histórica
 - Fundamentos (400 a.c.!).
 - Primeros computadores (sobre 1940).
 - Génesis (1943-1956).
 - Entusiasmo inicial, grandes esperanzas (1952-1969).
 - Crisis / redimensionamiento de problemas (1966-1973).
 - La industria de la IA (1980 hasta el presente).
- Seminario de Darmouth en 1956 como origen del término
- Sistemas basados en el conocimiento y sistemas basados en el aprendizaje



Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

Unidad 1 Lección 2

Resolución de problemas mediante búsqueda



Contenidos

1. Resolución de problemas mediante la abstracción
2. Formulación de problemas como espacios de estados
3. Árboles de búsqueda y búsqueda de soluciones

Objetivos

- Definir un problema desde el punto de vista computacional
- Buscar las acciones que lleven a la solución

Problemas y abstracción

- La abstracción es una pieza fundamental en diversas técnicas y métodos de la informática en general

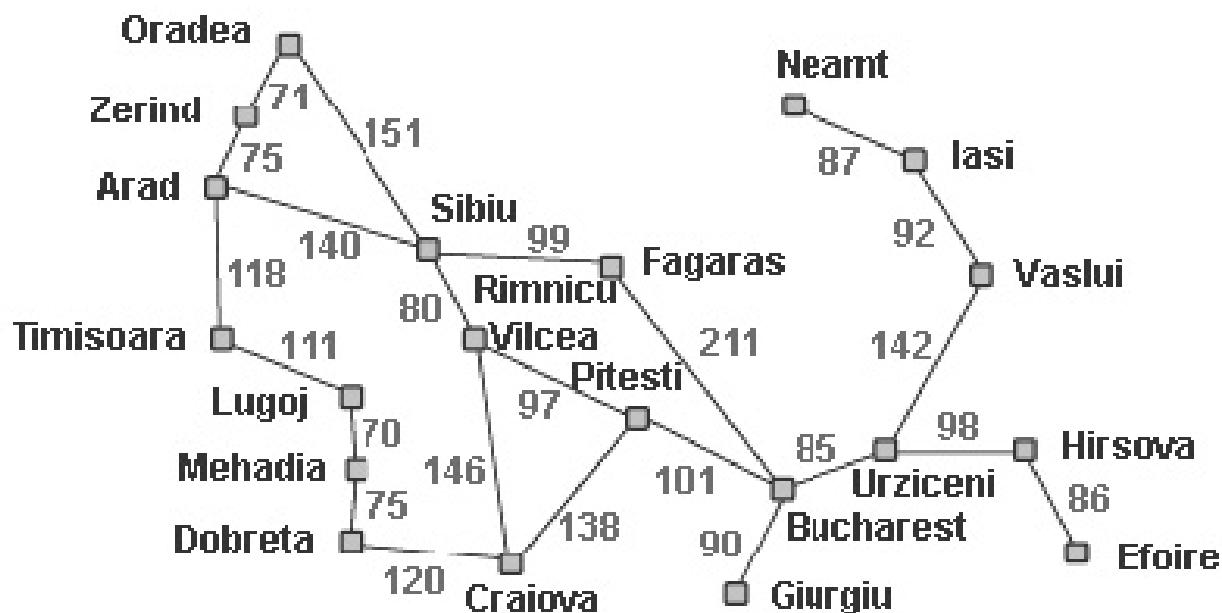


Por abstraer solemos entender el centrarnos sólo en los aspectos de un problema que identificamos como principales, sabiendo dejar aparte multitud de detalles que decidimos que no son relevantes.

- Caso de ejemplo: queremos ir de una ciudad a otra en Rumania (de Arad a Bucarest)

Problemas y abstracción

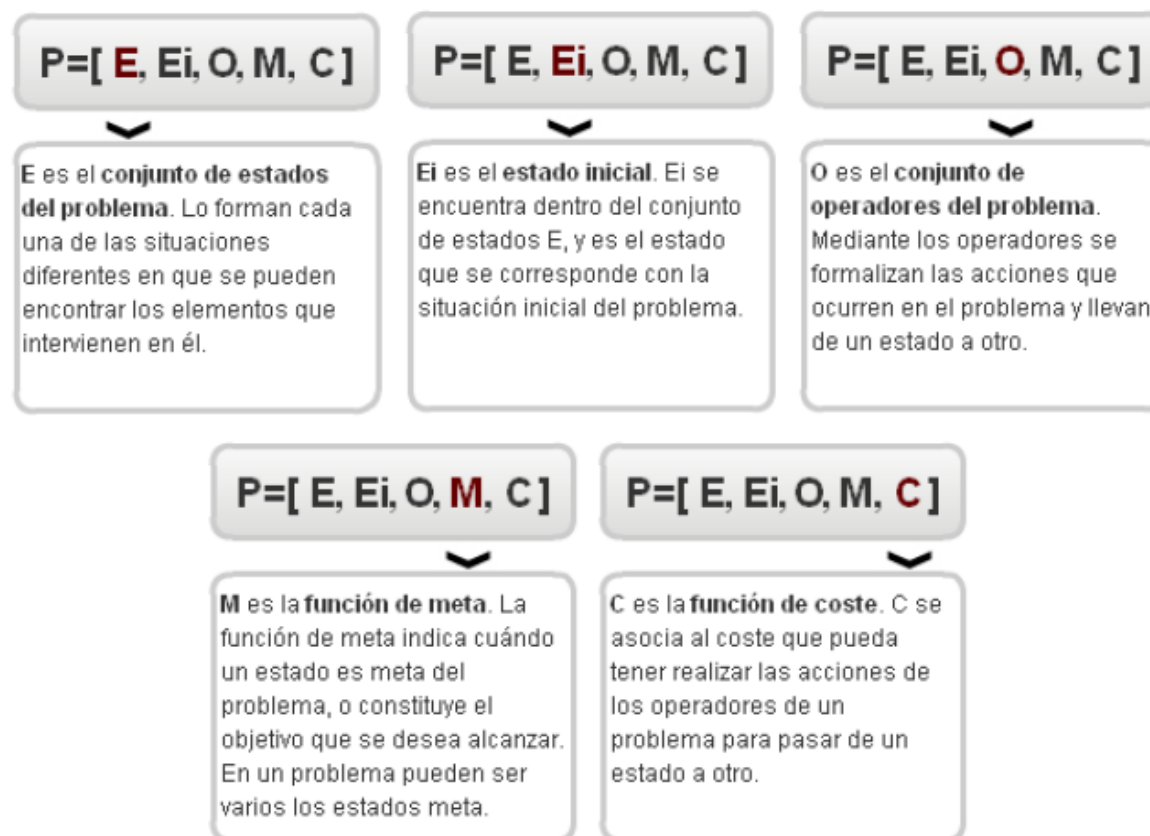
Recorrido por carreteras



AIMA (Russel & Norvig)

Formulación del problema como un espacio de estados

- Existen diversas formas de formalización, como puede ser una quintupla: $P = [E, E_i, O, M, C]$.

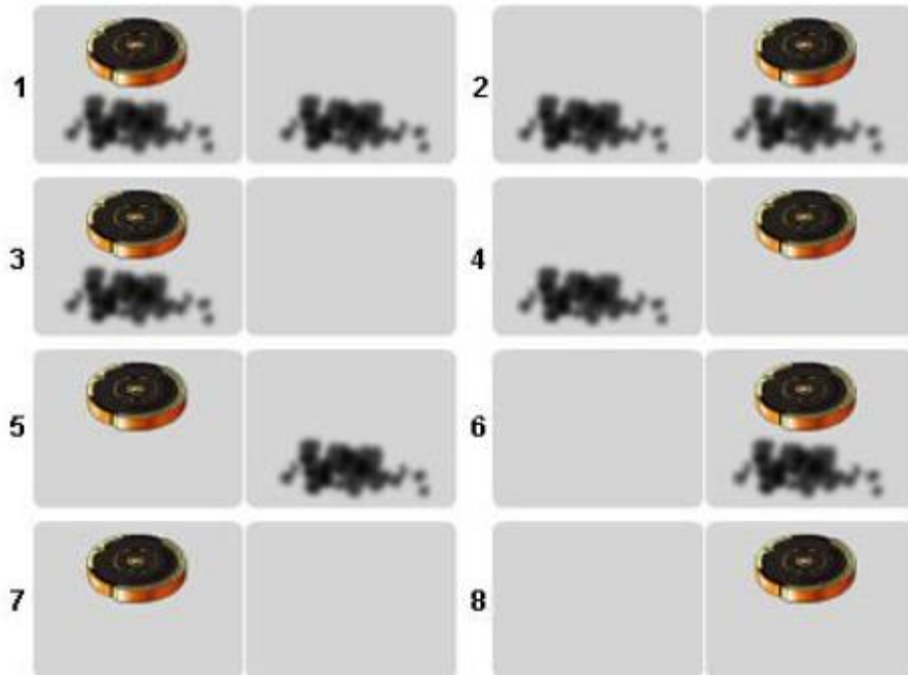


Formulación del problema como un espacio de estados

Espacio de estados	El espacio de estados de un problema son todos los estados alcanzables desde E_i aplicando los operadores del problema. El diagrama del espacio de estados de un problema es el grafo formado por nodos que son los estados del problema, unidos por arcos que son las acciones u operadores que llevan de uno a otro.
Camino	Un camino en un espacio de estados es una secuencia de estados conectados por una serie de acciones.
Solución de problemas	Definimos como solución de un problema , un camino que lleva del estado inicial E_i a un estado meta M . Típicamente un problema puede tener varias soluciones. También, ya que una solución es un camino, tiene definido un coste, que es el del camino en cuestión. La calidad de una solución puede venir determinada por su coste.

Ejemplo: robot aspirador (I)

- Construir un sistema de control de un aspirador, que tiene un motor para aspirar y otro para irse desplazando de un lugar a otro.

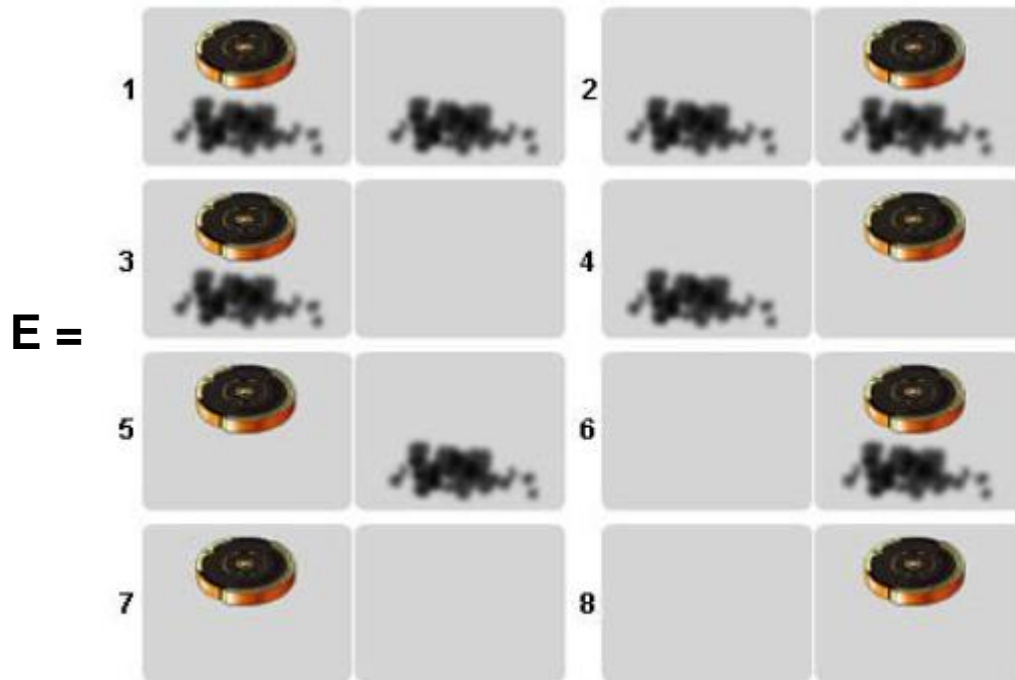


¿Cuántos posibles estados en un espacio de dos unidades donde cada una puede estar sucia o limpia?

2 posiciones robot * 2 posibles estados para celda1 * 2 posibles estados para celda2 = 8

Ejemplo: robot aspirador (II)

$P = [E, E_i, O, M, C]$



AIMA (Russel & Norvig)

$E_i = \{E1\}$

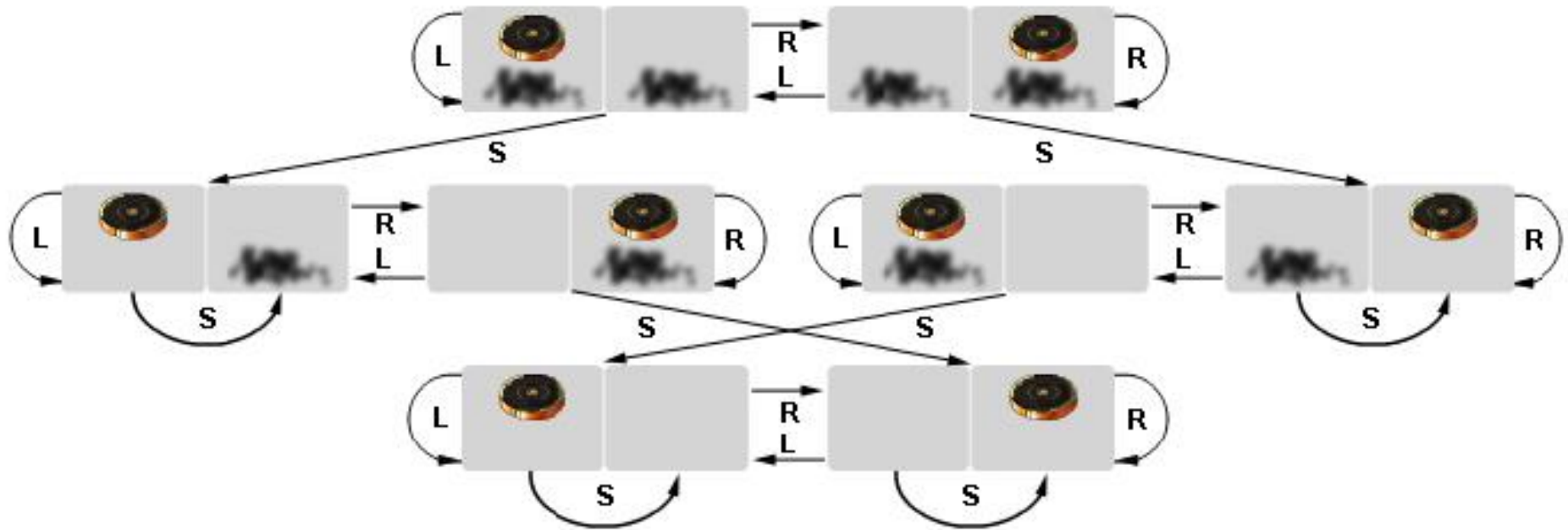
$O = \{O-I, O-D, O-S\}$

$M = \{E7, E8\}$

$C = \{1, 1, 1\}$

Ejemplo: robot aspirador (III)

Diagrama del espacio de estados del problema del robot aspirador



AIMA (Russell & Norvig)

¿Cuál es el coste mínimo?

¿Cuál es el coste máximo?

Ejemplo II: el puzzle (I)

Tablero del problema del puzle de 8 placas. Posición inicial y posición objetivo

5	4	
6	1	8
7	3	2

1	2	3
8		4
7	6	5

AIMA (Russell & Norvig)

Formalización en quintuplas: [E, Ei, O,M, C]

¿Cuántos posibles estados?

181.440 posibles estados

Ejemplo II: el puzzle (II)

- $E = \{E_1, \dots, E_n\} : n = 9!/2$ (la mitad de los estados son espejo)

- $E_i = \{E_1\}$

Diferentes modos de definir un operador

- $O = \{OA(1), OA(2), \dots, OD(1), \dots, OI(1), \dots, OAb(1), \dots, OAb(8)\}$

- $O = \{OA(h), OD(h), OI(h), Oab(h)\}$

Mover las piezas

Mover el hueco

- $M = \{E_2\}$

- $C = \{1, 1, \dots, 1\}$

Árboles de búsqueda y búsqueda de soluciones (I)

- Para resolver el problema es necesario disponer de un algoritmo que permita transitar por los diferentes estados para llegar del estado inicial al estado final deseado
- Este algoritmo se basa en una estructura de árbol para realizar la búsqueda y por eso denomina árbol de búsqueda
- Se ejecutan los operadores en el estado inicial y se obtienen los sucesores. Posteriormente se aplican otra vez los operadores con éstos y así sucesivamente hasta encontrar la meta

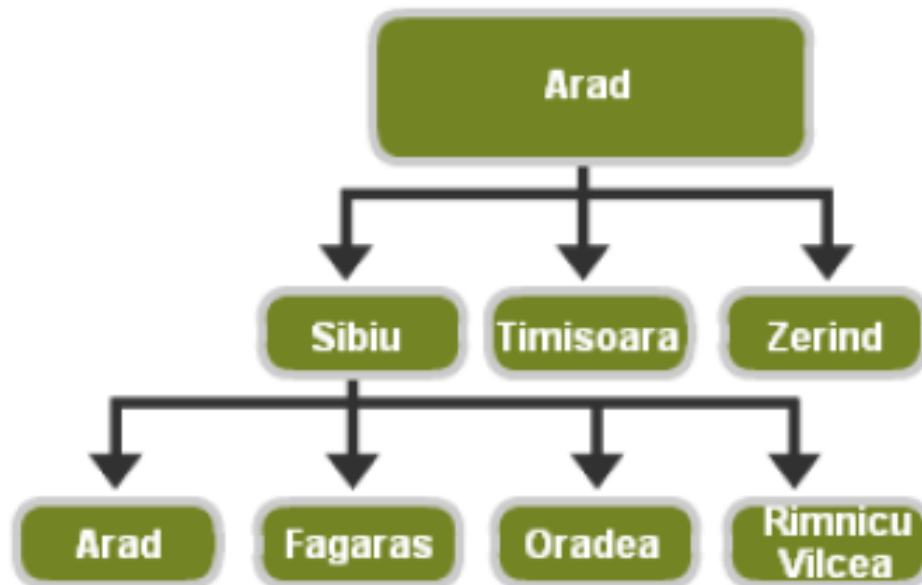
Árboles de búsqueda y búsqueda de soluciones (II)

Pasos en la creación de un árbol de búsqueda

(a) Estado inicial

(b) Después de expandir

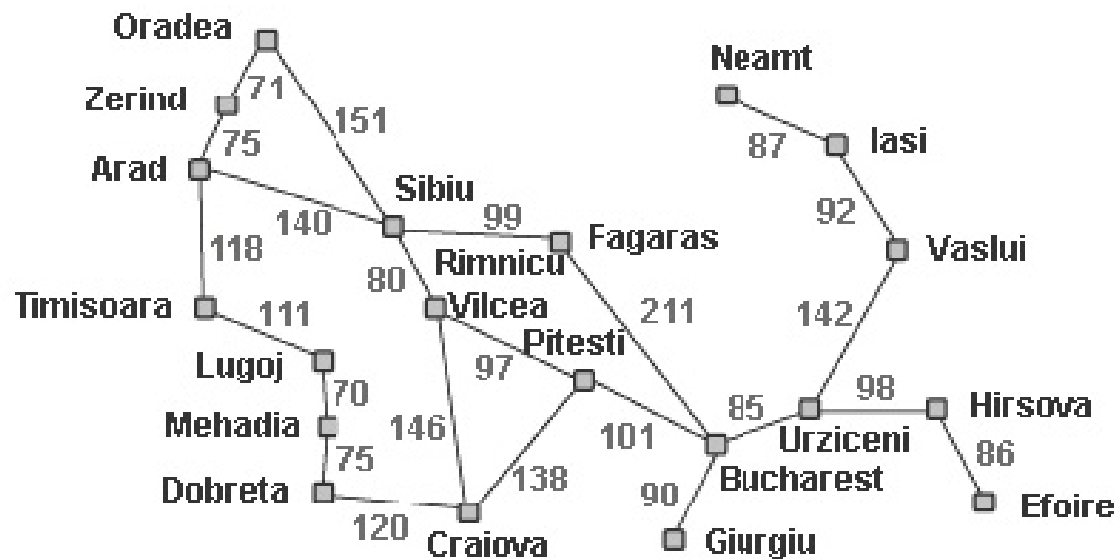
(c) Después de expandir Sibiu



Árboles de búsqueda y búsqueda de soluciones (III)

- El viajero quería ir de Arad a Bucarest
- **Una solución** es [Arad, Zerind, Oradea, Sibiu, Fagaras, Bucarest]

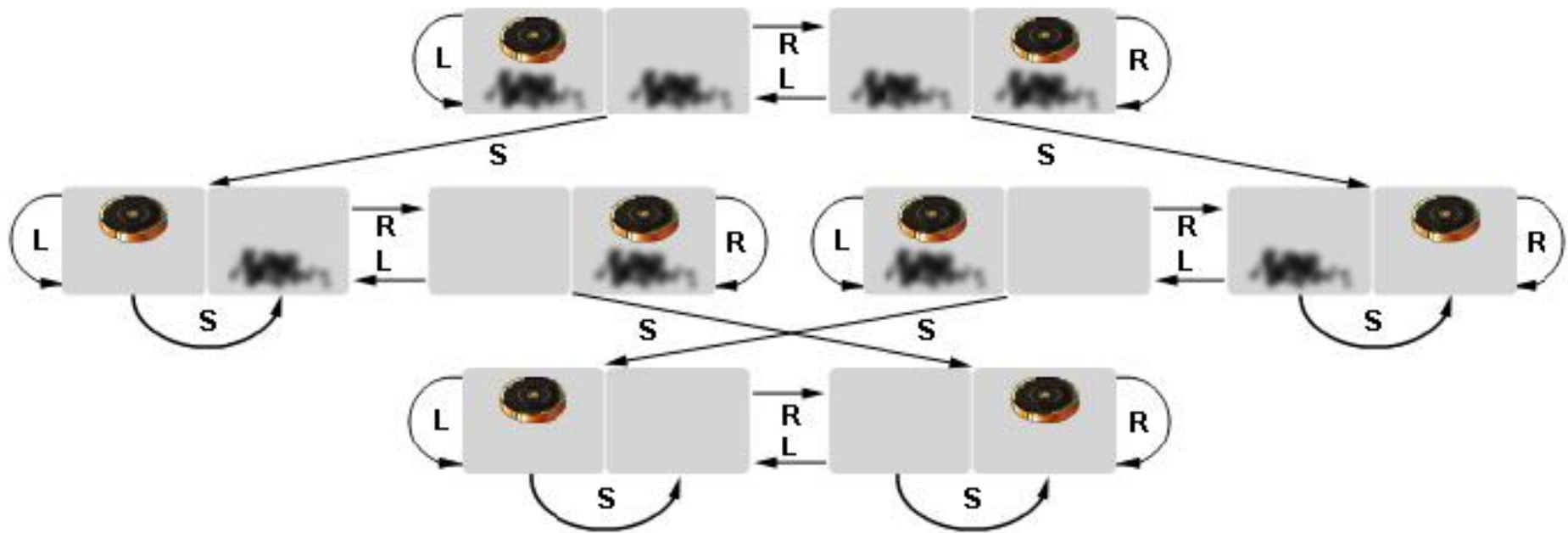
Recorrido por carreteras



AIMA (Russel & Norvig)

Árboles de búsqueda y búsqueda de soluciones (IV)

Diagrama del espacio de estados del problema del robot aspirador



AIMA (Russell & Norvig)

Árboles de búsqueda y búsqueda de soluciones (V)

Función búsqueda-árbol **devuelve** una solución o fallo

bucle hacer

si no hay candidatos para expandir entonces devolver fallo
escoger, de acuerdo a la estrategia, un nodo hoja para expandir
si el nodo contiene un estado objetivo entonces devolver la
correspondiente solución

en otro caso expandir el nodo y añadir los nodos resultado al
árbol de búsqueda

final bucle

Final función

¿Cual es el orden de expansión del árbol? ¿Qué tipo de árbol se genera?

Resumen

- Técnicas de modelado de problemas a través de la abstracción $P = [E, E_i, O, M, C]$
 - **E**: conjunto de estados del problema.
 - **E_i**: estado inicial.
 - **O**: conjunto de operadores del problema.
 - **M**: función de meta.
 - **C**: función de coste.
- Métodos de resolución de problemas a través de la búsqueda de caminos
 - Árbol de búsqueda como algoritmo para obtener la solución



Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

Unidad 1

Lección 3

Estrategias de búsqueda



Contenidos

1. Métodos de búsqueda
2. Búsqueda en anchura
3. Búsqueda por coste uniforme
4. Búsqueda en profundidad
5. Búsqueda en profundidad limitada
6. Búsqueda en profundidad iterativa
7. Búsqueda bidireccional
8. Comparativa entre búsquedas

Objetivos

- Distinguir las diferentes estrategias de búsqueda de soluciones
- Utilización de los algoritmos de cada una de ellas
- Elaborar el árbol de búsqueda
- Evaluar su rendimiento en base al tiempo y la memoria que necesitan

Métodos de búsqueda (I)

- El objetivo es conseguir un método o algoritmo que permite ir desde el estado inicial al estado meta o final de con el menor coste posible
- Seis métodos diferentes que representan las diferentes posibilidades de búsqueda

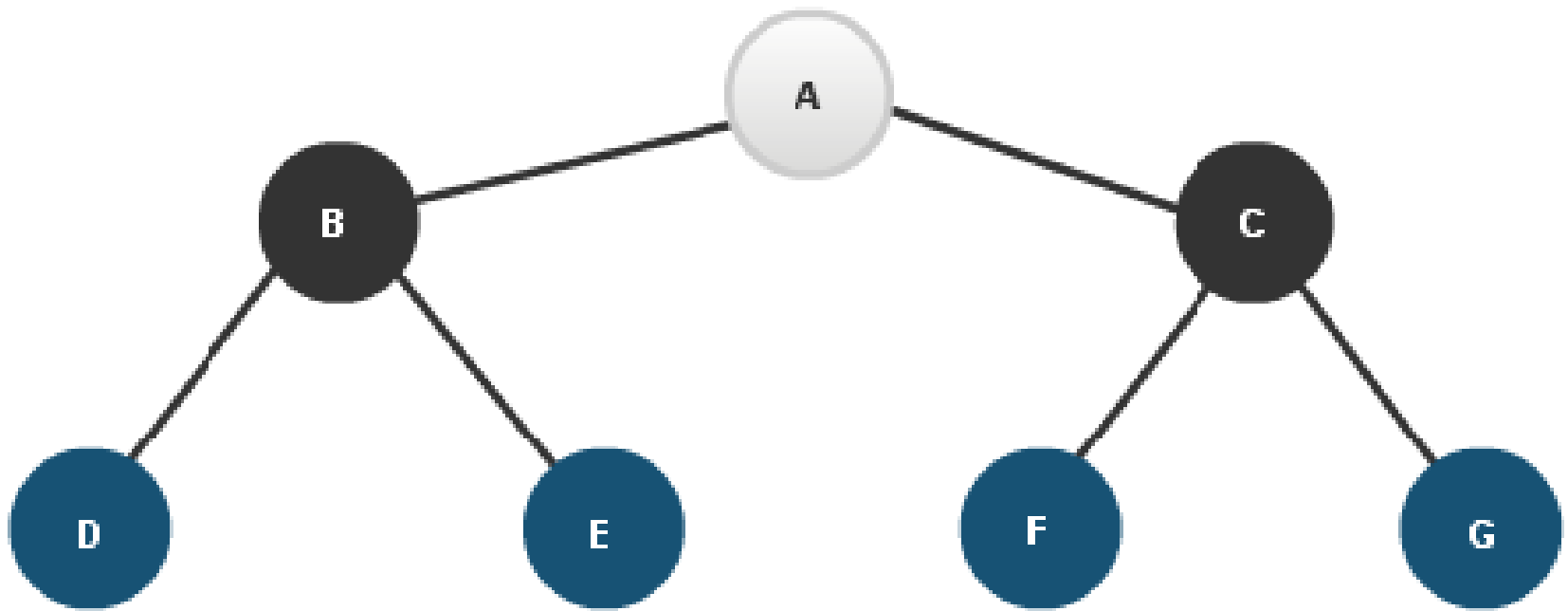
Métodos de búsqueda (II)

- Para medir el rendimiento del método solución:

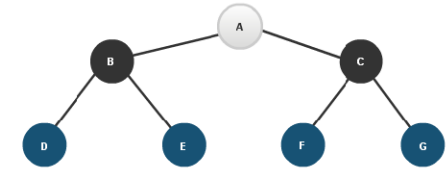
Complejidad	Si el método es completo, está garantizado que si el problema tiene una solución, el método la encuentra.
Optimización	Si la solución encontrada garantiza el menor coste C tiene.
Complejidad en el tiempo	Es el tiempo que tarda directamente en encontrarla.
Complejidad en el espacio	Es el espacio que ocupará dependerá del tamaño de la parte del árbol que sea necesario mantener en memoria para encontrar la solución.

- **El factor de ramificación, b .** Está determinado por el número medio de sucesores o hijos de cada nodo.
- **La profundidad del nodo, d .** Objetivo meta más superficial.
- **La longitud máxima de cualquier camino en el espacio de estados, m .** Es la profundidad máxima del árbol de búsqueda.

Búsqueda en anchura (I)



Búsqueda en anchura (II)



Comportamiento del método

Si revisamos esta forma de construcción del árbol de búsqueda desde el punto de vista de los cuatro aspectos que mencionamos en el punto anterior, observamos lo siguiente:

- **Complejidad:** la búsqueda primero en anchura sí es completa, y está garantizado que si el problema tiene una solución, este método la encuentra.

- **Optimización**

profundidad

se encuentre

menor costo

- **Complejidad**

tamaño del

	Profundidad	Nodos	Tiempo	Memoria
	2	1.100	11 segundos	1 megabyte
	4	111.100	11 segundos	106 megabytes
	6	10^7	19 minutos	10 gigabytes
	8	10^9	31 horas	1 terabytes
	10	10^{11}	129 días	101 terabytes
	12	10^{13}	35 años	10 petabytes
	14	10^{15}	3.523 años	1 exabyte

- **Complejidad en espacio:** el espacio en memoria que demanda este método es también del orden del tamaño del árbol, $O(b^d)$.

Búsqueda de coste uniforme (I)

- Utiliza el coste de los operadores ejecutados para pasar de un estado a otro como medida de guiado durante la búsqueda
- Se expande el nodo que tiene un menor coste desde la raíz

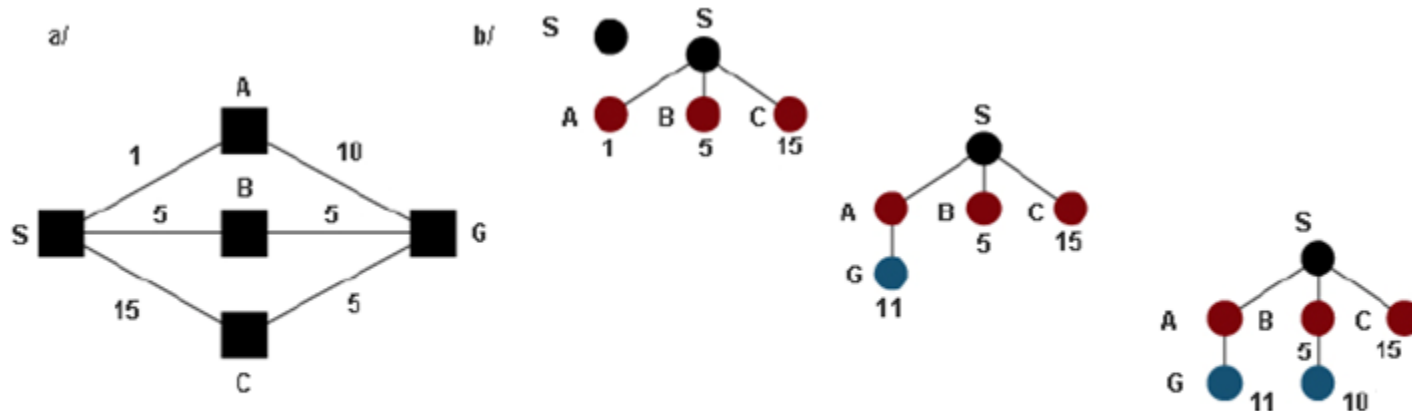


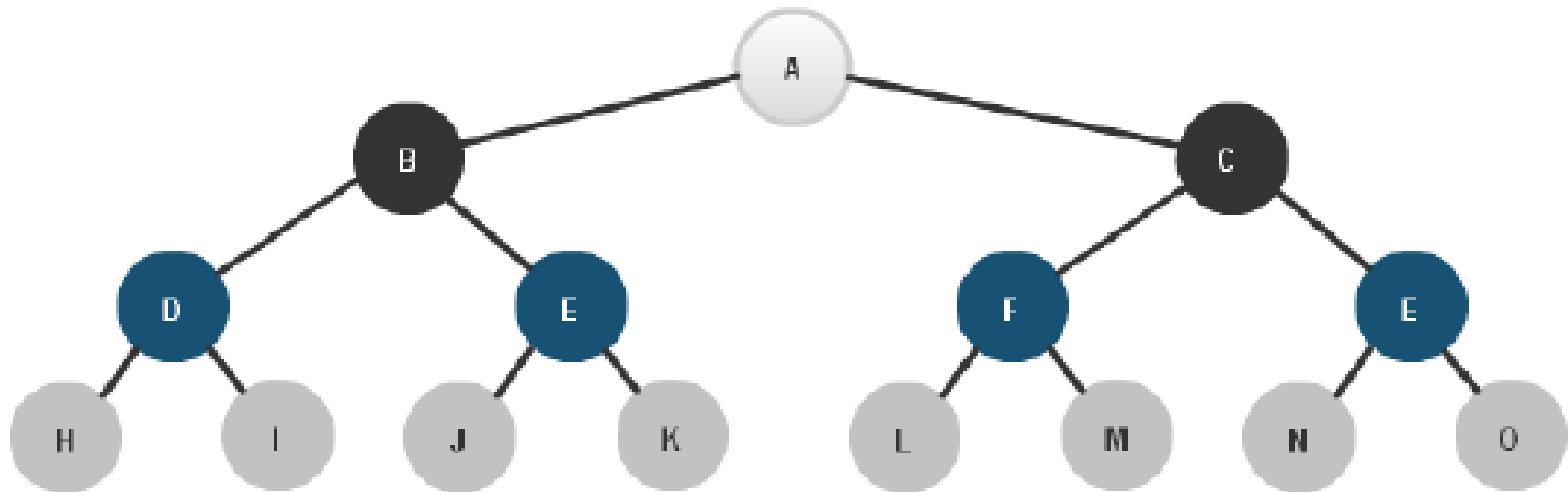
Diagrama de espacio de estados de un problema, y construcción del árbol por coste uniforme

Solución: {S,B,G}

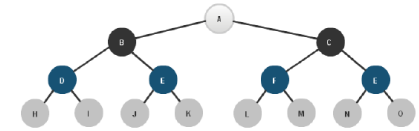
Búsqueda de coste uniforme (II)

- Completitud: se puede garantizar completitud si el costo de cada paso es mayor o igual a alguna constante positiva pequeña ε
- Optimización: garantiza el encontrar, entre las diferentes soluciones, aquella que conlleva menor coste de ruta g , calculado como la suma de los costes C de las transiciones de estado
- Complejidad en el tiempo: en el peor caso $O(b^c)$ donde puede que $c \gg d$, si el coste es constante entonces $O(b^d)$
- Complejidad en el espacio: es de $O(b^d)$

Búsqueda en profundidad (I)



Búsqueda en profundidad (II)



- Completitud: esta solución no es completa ya que no garantiza que se encuentre la solución en el caso de que alguna de las ramificaciones sea infinita (o recursiva)
- Optimización: la solución no es la óptima ya que no garantiza encontrar la que tenga un menor número de niveles de profundidad ni tampoco que los costes de los operadores aplicados sean los mínimos
- Complejidad en el tiempo: en el peor caso $O(b^m)$ (Nótese que m puede ser mucho mayor que d)
- Complejidad en el espacio: es de $O(bm)$ (siendo b el factor de ramificación)

Búsqueda en profundidad limitada (l)

- Se aborda el problema de posibles ramas infinitas estableciendo un límite máximo l de profundidad para la búsqueda de soluciones
 - Si la solución está a una profundidad d siendo $d > l$ entonces no se encontrará
 - Puede devolver dos casos: i) fracaso, la solución no existe, o ii) corte, la solución no se encuentra en el límite establecido por l pero puede existir

Búsqueda en profundidad limitada (II)

- Completitud: esta solución no es completa en general; sí lo es si tomamos $l > d$.
- Optimización: la solución no es la óptima ya que no garantiza encontrar la que tenga un menor número de niveles de profundidad ni tampoco que los costes de los operadores aplicados sean los mínimos
- Complejidad en el tiempo: en el peor caso $O(b^l)$ (Nótese que m es mucho mayor que d)
- Complejidad en el espacio: es de $O(bl)$ (siendo b el factor de ramificación)

Búsqueda en profundidad iterativa (I)

- El límite l asociado a la profundidad máxima del espacio de búsqueda del árbol se incrementa iterativamente $l=0, l=1, l=2, l=3, \dots, l = m$
- El límite se incrementa hasta que se encuentra la solución
- Este tipo de búsqueda es más derrochadora ya que algunos estados son visitados varias veces (los nodos de nivel d son generados 1 vez, los de $d-1$ 2, los de $d-2$ 3, y así sucesivamente hasta la raíz que son visitados d veces)

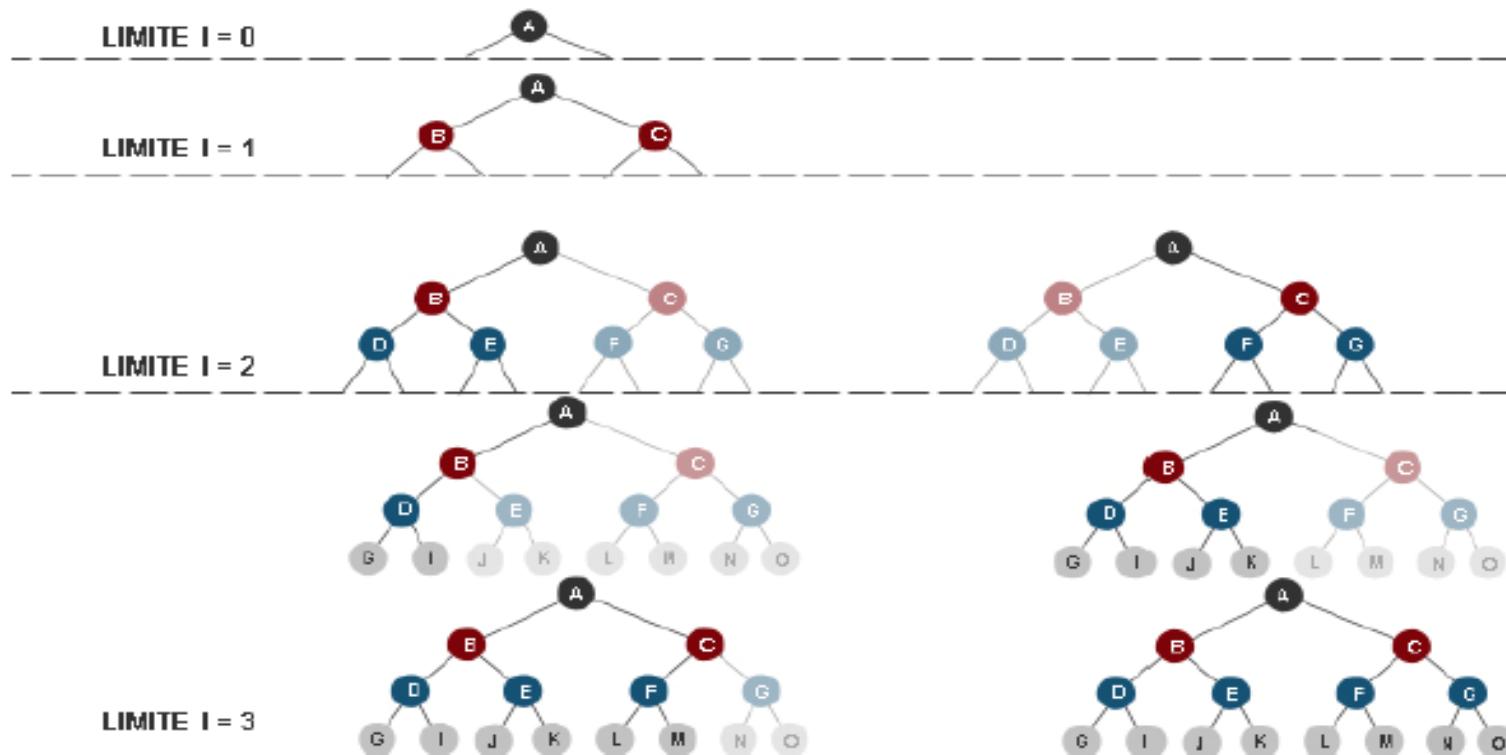
Búsqueda en profundidad iterativa (II)

- El límite l asociado a la profundidad máxima del espacio de búsqueda del árbol se incrementa iterativamente $l=0, l=1, l=2, l=3, \dots, l = m$
- El límite se incrementa hasta que se encuentra la solución
- Combina las ventajas de la búsqueda en profundidad (memoria) y en anchura (completitud y óptima si el coste no disminuye con la profundidad del nodo)
- Este tipo de búsqueda es más derrochadora ya que algunos estados son visitados varias veces (los nodos de nivel d son generados 1 vez, los de $d-1$ 2, los de $d-2$ 3, y así sucesivamente hasta la raíz que son visitados d veces)

Búsqueda en profundidad iterativa (III)

- Completitud: sí es completa, si existe solución se encuentra
- Optimización: la solución no es la óptima ya que no garantiza encontrar la que tenga un menor número de niveles de profundidad ni tampoco que los costes de los operadores aplicados sean los mínimos
- Complejidad en el tiempo: en el peor caso $O(b^d)$
- Complejidad en el espacio: es de $O(bd)$ (siendo b el factor de ramificación)

Búsqueda en profundidad iterativa (IV)



Si $b=2$ y $d=3$, el número de nodos generados es $6 + 8 + 8 = 22$

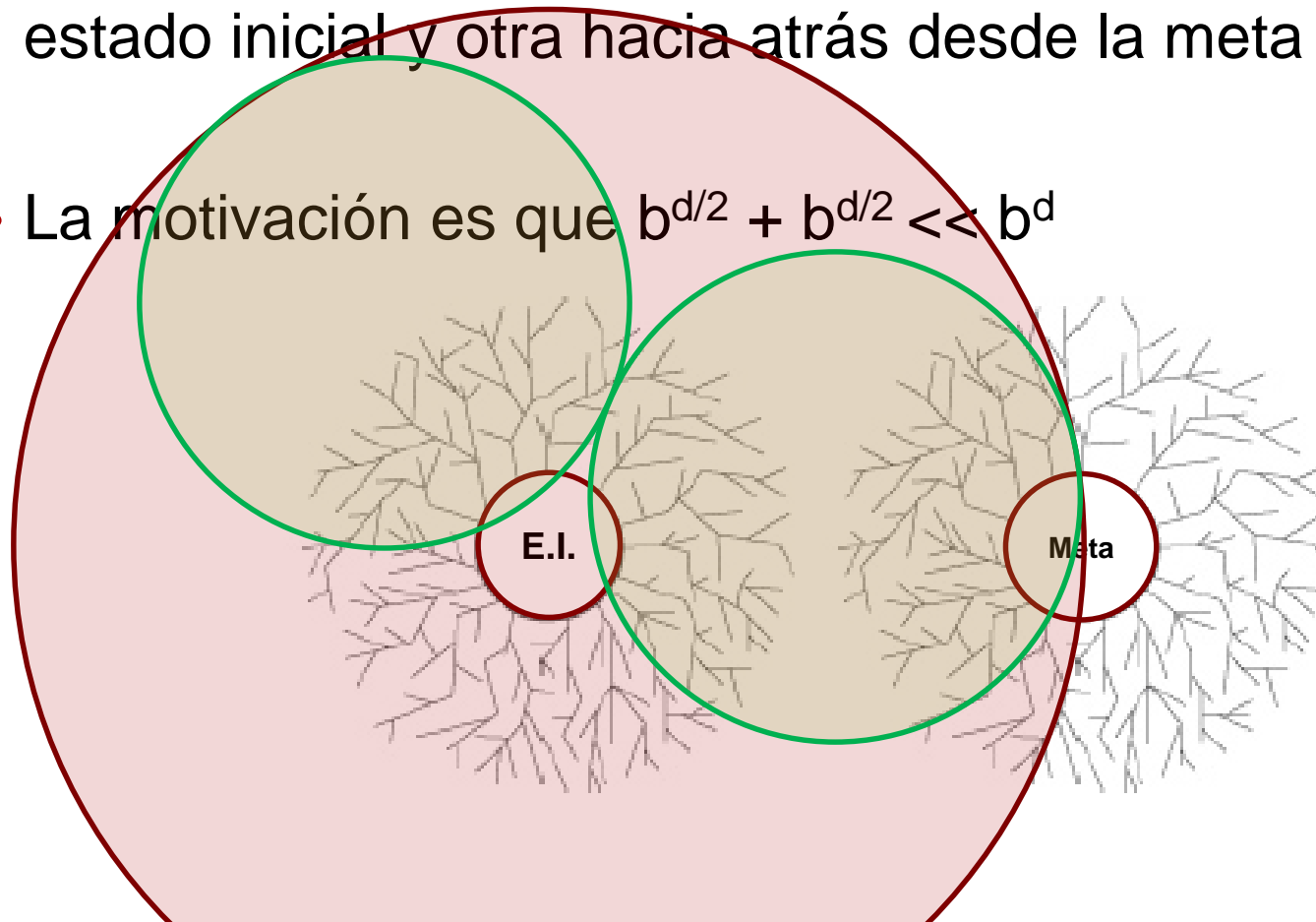
Si $b=10$ y $d=5$, el número de nodos generados es $50+400+3000+20000+100000=123450$

Búsqueda en profundidad iterativa (V)

- En general la búsqueda en profundidad iterativa es el método de búsqueda sin conocimiento preferido cuando hay un espacio de búsqueda grande y no se conoce la profundidad de la solución

Búsqueda bidireccional (I)

- La idea de la búsqueda bidireccional es ejecutar dos búsquedas simultáneas: una hacia adelante a partir del estado inicial y otra hacia atrás desde la meta
- La motivación es que $b^{d/2} + b^{d/2} \ll b^d$



Búsqueda bidireccional (II)

La búsqueda bidireccional aplicada en problemas concretos

No en todos los problemas puede utilizarse, ya que es necesario que puedan calcularse los inversos de los operadores. En el problema del **recorrido de ciudades**, por ejemplo, sí puede utilizarse: habría que construir un árbol de búsqueda que parte de la ciudad origen, y otro, simultáneamente, que parte de la ciudad destino; cuando los dos árboles llegan a tener un nodo común y se encuentran, se ha conseguido la solución. En el problema de las **ocho placas**, por ejemplo, también puede utilizarse.

Búsqueda bidireccional (III)

- Completitud: sí es completa, si existe solución se encuentra
- Optimización: depende del método de exploración
- Complejidad en el tiempo: en el peor caso $O(b^{d/2})$
- Complejidad en el espacio: es de $O(b^{d/2})$

Búsqueda bidireccional (IV)

Ejemplo numérico de la búsqueda bidireccional

Un ejemplo numérico sería: si $b = 10$ y $d = 6$, pasamos de $10^6 = 1.000.000$ a $10^3 = 1.000$.

Es decir, si construir un árbol tiene un coste 1.000, construir los dos árboles $2 \times 1.000 = 2.000$, e incluir el procedimiento de detección de estados comunes $2 \times 2.000 = 4.000$. Es decir nos mantenemos en ese tipo de magnitud, en contraste con el 1.000.000 utilizando sólo un árbol.

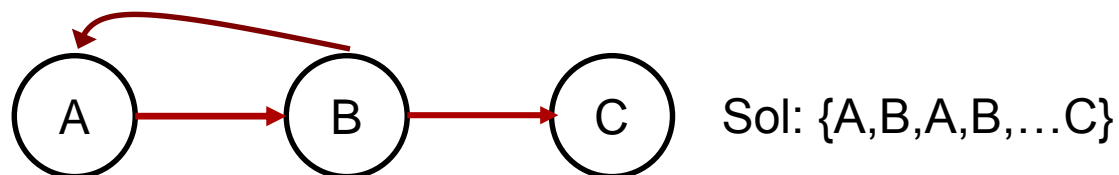
Comparación de estrategias

Criterio	Anchura	Coste Uniforme	Profundidad	Profundidad limitada	Profundidad iterativa	Bidireccional
Compleitud	SI	SI	NO	SI	SI	SI
Optimalidad	SI	SI(coste)	NO	NO	SI	SI
Tiempo	$O(b^d)$	$O(b^c)$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Espacio	$O(b^d)$	$O(b^d)$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$

¿Qué hacer cuando existen estados repetidos?


Evitar estados repetidos

- Se producen estados repetidos o ciclos en el camino de búsqueda cuando las acciones son reversibles (p.e. problemas de búsqueda en rutas, o el puzzle)



- Si no se controlan los estados repetidos un problema resoluble se puede convertir en irresoluble
- La búsqueda con limite en profundidad soluciona este problema aunque no lo hace de manera explícita

Evitar estados repetidos

- Se utilizan listas para almacenar el conjunto de nodos del árbol que ya han sido visitados, si un nodo ya ha sido visitado se ignora salvo que sea de menor coste que el previo
- A esto se le denomina búsqueda en grafos
- Los casos de búsqueda de coste uniforme y la búsqueda en anchura con coste uniforme son también óptimas para búsquedas en grafos
- Mantener en memoria los caminos recorridos aumenta las restricciones de su uso para algunos tipos de búsqueda 

Resumen

- Dado un modelado hemos visto como ejecutar un método de resolución basado en búsqueda
 - Búsqueda en anchura
 - Búsqueda por coste uniforme
 - Búsqueda en profundidad
 - Búsqueda en profundidad limitada
 - Búsqueda en profundidad iterativa
 - Búsqueda bidireccional
- Modos de medir su rendimiento
 - Completitud
 - Optimización
 - Complejidad en el tiempo
 - Complejidad en el espacio
- Evitar estados repetidos



Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

Unidad 1

Lección 4

Métodos de búsqueda basados en información




Contenidos

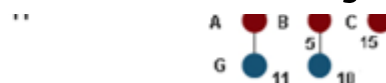
1. Búsqueda primero el mejor
2. Búsqueda avariciosa (greedy approach)
3. Búsqueda A*
4. Funciones heurísticas
5. Evaluación experimental
6. Diseño de heurísticas

Objetivos

- Utilizar heurísticas para mejorar el rendimiento en la búsqueda en el espacio de soluciones
- Aprender a utilizar el método primero el mejor
- Aprender a utilizar la búsqueda avariciosa y A*

Búsqueda primero el mejor (I)

- Trata de elegir el nodo que “parece” el mejor
- Utiliza una función de evaluación f_e que proporciona para un nodo n información sobre lo deseable que es para llegar a la meta: $f_e(n)$
- El nodo elegido no es siempre la mejor opción! 
- La función heurística $h(n) =$ coste estimado del camino mas barato desde el nodo n a un nodo objetivo

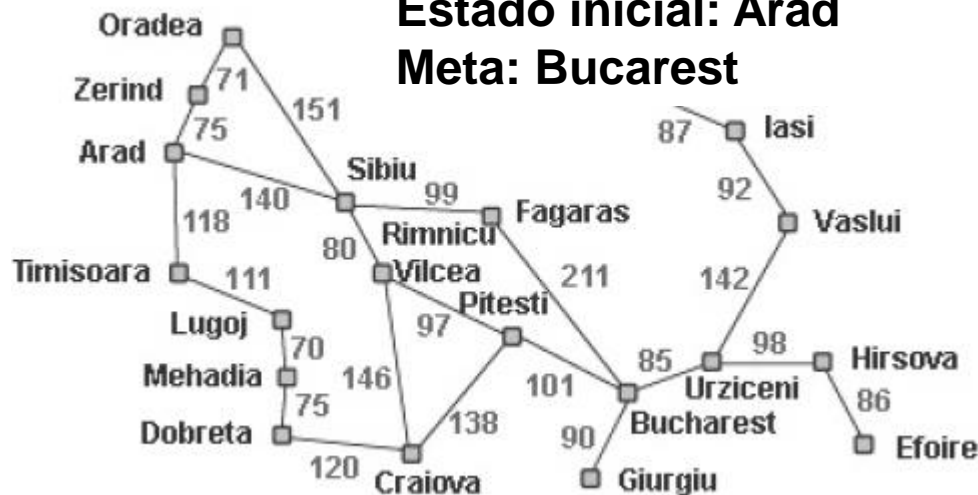


¿Cuál sería una heurística en el caso del viajero en Rumania?

Búsqueda avariciosa (I)

- Se trata de un caso específico de la búsqueda primero el mejor
- Trata de expandir el nodo que “parece” ser el más cercano a la meta $f_e(n) = h(n)$

Objetivo: ir desde Arad a Bucarest
Estado inicial: Arad
Meta: Bucarest

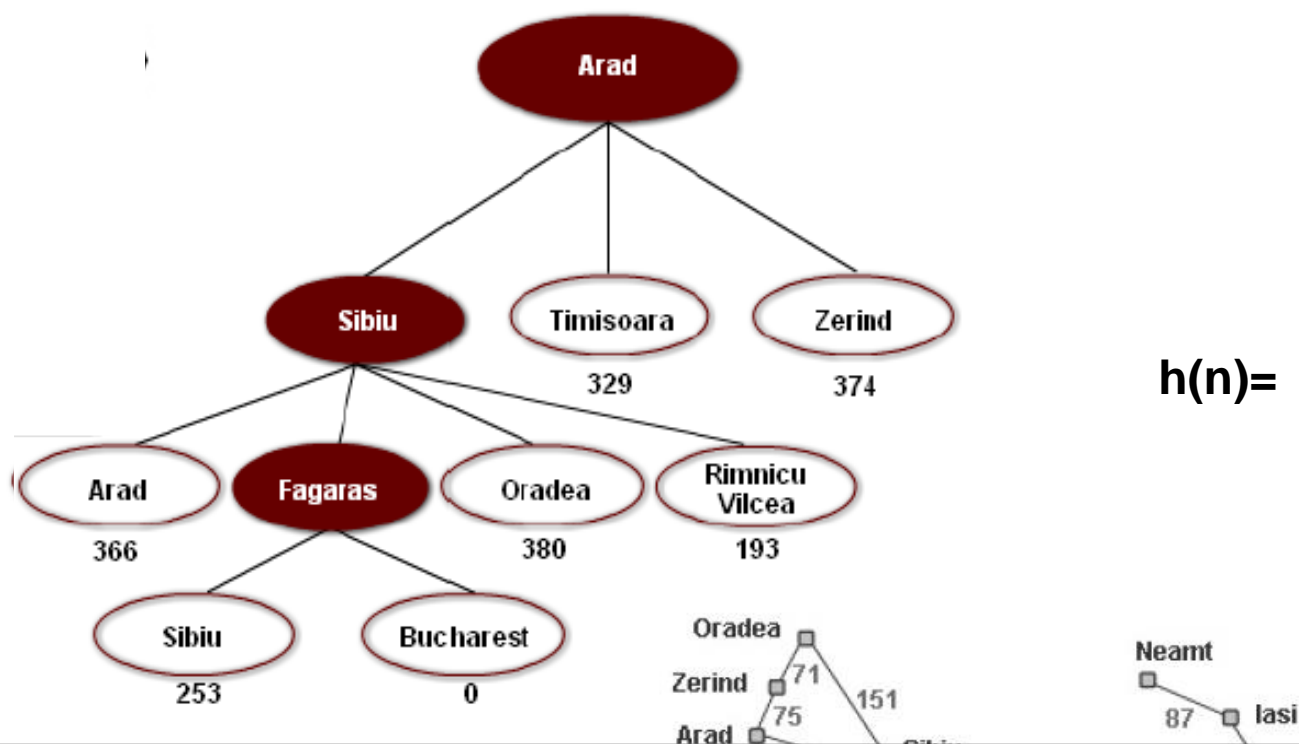


$h(n)=$

Distancia en
línea recta a Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Búsqueda avariciosa (II)



$g([Arad, Sibiu, Fagaras, Bucarest]) = 450$



$g([Arad, Sibiu, Rimnicu Vilcea, Pitesti, Bucarest]) = 418$

Búsqueda avariciosa (III)

- Completitud: no es completa, no se garantiza que si el problema tiene solución el método la encuentre. Si se controlan los ciclos repetitivos entonces sí.
- Optimización: no se garantiza que la solución obtenida sea la de menor coste.
- Complejidad en el tiempo: en el peor caso $O(b^d)$

Búsqueda A* (I)

- Es la búsqueda más conocida de entre los métodos de primero el mejor
- Auna las ventajas de la búsqueda por coste uniforme (completitud y optimalidad) con las de la búsqueda avariciosa (velocidad incorporando heurísticas)
- Sigue el método de búsqueda de primero el mejor incorporando la heurística a los valores de coste conocidos de los operadores: $f_e(n) = g(n) + h(n)$

Búsqueda A* (II)

- Completitud: sí es completa, se garantiza que si el problema tiene solución el método la encuentre.
- Optimización: sí se garantiza que la solución obtenida sea la de menor coste.
- Complejidad en el tiempo: en el peor caso $O(b^d)$, pero en general es el más rápido de los vistos junto con la búsqueda avariciosa
- Complejidad en el tiempo: en el peor caso $O(b^d)$

Búsqueda A* (III)

Búsqueda por coste uniforme: $f_e(n) = g(n)$. La búsqueda por coste uniforme trabaja con una f_e calculada "mirando sólo hacia atrás" para seleccionar el nodo a expandir (g).

Búsqueda avariciosa: $f_e(n) = h(n)$. La búsqueda avariciosa "sólo mira hacia adelante" al utilizar $h(n)$.

Búsqueda A*: $f_e(n) = g(n) + h(n)$. La búsqueda A* es la más inteligente, al utilizar para el cálculo de la función de evaluación lo recorrido hasta el momento (g) y la estimación de lo que queda por delante (h).

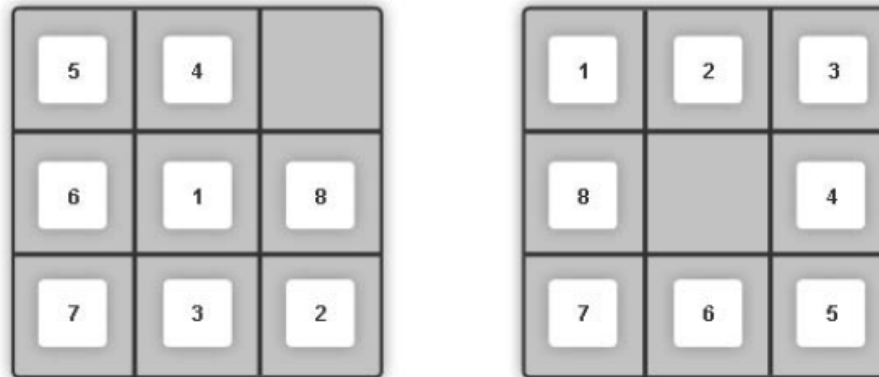
Las heurísticas utilizadas generalmente son optimistas y se considera admisible siempre que no sobreestime el coste real (sino no se cumpliría la optimalidad del método)

Funciones heurísticas (I)

- ¿Cómo conseguir una buena función heurística?
 - Bibliografía, proyectos similares, análisis del problema, etc.
- Existen un conjunto de elementos comunes a cualquier función heurística durante su definición, diseño, integración y evaluación

Funciones heurísticas (II)

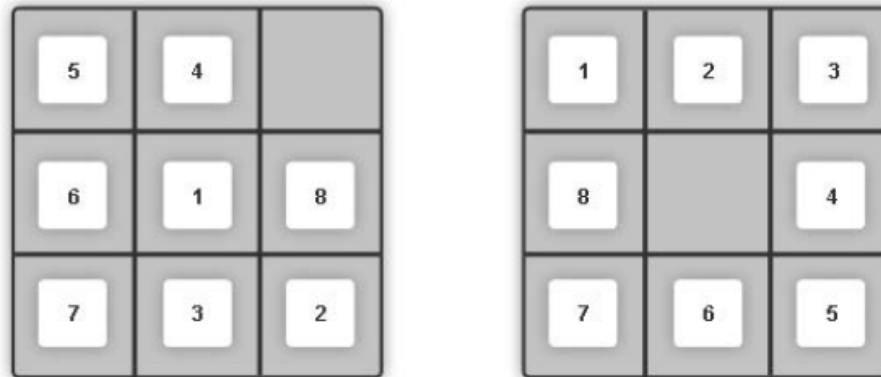
- Problema de las ocho placas.



- La heurística debe proporcionar el número de movimientos que quedan por realizar para llegar a la meta
- Tres heurísticas que podemos definir para este problema son las que siguen:
 - $h^0(n) = 0$
 - $h^1(n)$ = número de placas que están en lugar incorrecto en el estado n
 - $h^2(n)$ = suma de las distancias (vertical + horizontal) que separan a las placas de su posición meta

Funciones heurísticas (III)

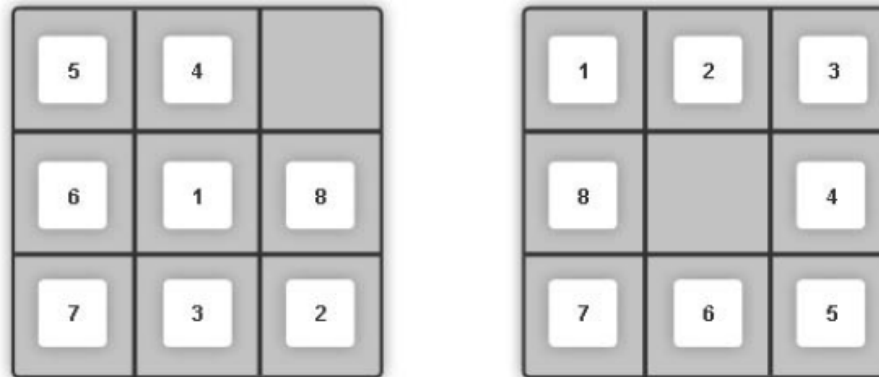
- Problema de las ocho placas.



- Los valores que se obtienen para esta heurística son:
 - $h^0(n) = 0$, como para cualquier n .
 - $h^1(n) = 7$, que son las placas que no están en su lugar de destino.
 - $h^2(n) = 4+2+2+2+2+0+3+3=18$ (el 4 es la distancia que separa a la placa 5 de su posición meta).

Funciones heurísticas (IV)

- Problema de las ocho placas.



- Para todo nodo (n) estas tres heurísticas tienen las siguientes propiedades: $h^0(n) < h^1(n) < h^2(n)$, y por tanto:
 - h^0 es la que menor valor da siempre y es la más "optimista" (de hecho es la heurística trivial: supone que el coste de ruta hasta la meta es 0).
 - h^1 da un valor estimado más próximo al coste real.
 - h^2 es la que da el mayor valor y más próximo al coste de ruta real (para el caso de la figura que hemos visto el coste de ruta es 26).

Evaluación experimental (I)

- Para evaluar la calidad y el rendimiento de las heurísticas se puede recurrir a la evaluación experimental
- Un parámetro utilizado es el factor de ramificación b^*
- Si el número de nodos generados por un método de búsqueda para un problema particular es N , y la profundidad de la solución es d , entonces b^* debería tener la siguiente forma:

$$N \approx 1 + (b^*)^1 + (b^*)^2 + \dots + (b^*)^d$$

- Una heurística bien diseñada tendría un valor de b^* cercano a 1

Evaluación experimental (II)

- ¿Cómo se diseña la evaluación?
- Consideramos el problema de las ocho placas y las dos heurísticas h^1 y h^2
- Se ejecuta el algoritmo de búsqueda muchas veces (1200) con estados iniciales diferentes y elegidos de manera aleatoria para longitudes de profundidad de 2 a 24
- Se resuelve con la búsqueda de profundidad iterativa y con la búsqueda en árbol A^* usando tanto h^1 como h^2

Evaluación experimental (III)

Tabla Coste de búsqueda (N)

d	BPI	A*(h ₁)	A*(h ₂)
2	10	6	6
4	112	13	12
6	680	20	18
8	6.384	39	25
10	47.127	93	39
12	364.404	227	73
14	3.473.941	539	113
16	-	1.301	211
18	-	3.056	363
20	-	7.276	676
22	-	18.094	1.219
24	-	39.135	1.641

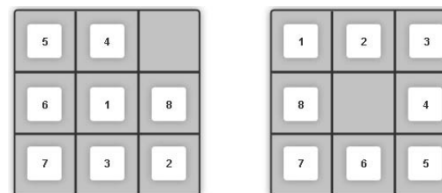
Tabla Factor de ramificación eficaz (b*)

d	BPI	A*(h ₁)	A*(h ₂)
2	2,45	1,79	1,79
4	2,87	1,48	1,45
6	2,73	1,34	1,30
8	2,80	1,33	1,24
10	2,79	1,38	1,22
12	2,78	1,42	1,24
14	2,83	1,44	1,23
16	-	1,45	1,25
18	-	1,46	1,26
20	-	1,47	1,27
22	-	1,48	1,28
24	-	1,48	1,26

$h^2(n) > h^1(n) \rightarrow$ decimos que h^2 domina sobre h^1 y nunca expandirá más nodos

Diseño de heurísticas (I)

- Problema relajado
 - Dado un problema P, un problema relajado incluye un subconjunto de las restricciones de P sobre sus operadores. Se cumple que si el coste del problema relajado es admisible entonces también lo es para la heurística de P



- En el caso de las ocho placas
 - Se puede mover una placa del cuadro A al cuadro B, si B está en el tablero
 - Se puede mover una placa del cuadro A al B si A está junto a B
 - Se puede mover una placa de A a B si B está vacío

Diseño de heurísticas (II)

- Composición por dominancia
 - Dadas varias heurísticas (h^2, h^1) podemos elegir la que presente el mejor comportamiento, es decir la que sea más eficiente
 - Se utiliza $h(n) = \max(h^2, h^1)$ en el caso de las ocho placas, o más genéricamente $h(n) = \max(h^i(n))$,
 - En el ejemplo de las ocho placas: $h^2 : h^2(n) > h^1(n)$

Resumen

- Como introducir en el problema el conocimiento a priori a través de funciones heurísticas
- Búsqueda primero el mejor: $f_e(n) = g(n)$
- Búsqueda avariciosa (greedy approach): $f_e(n) = h(n)$
- Búsqueda A^* : $f_e(n) = g(n) + h(n)$
- Diferentes pasos en la aplicación de heurísticas:
 - Evaluación de las heurísticas (factor de ramificación)
 - Diseño (problema relajado y comparación por dominancia)

Actividad

Objetivo

- Comprender cómo se generan los árboles de búsqueda de soluciones.

Enunciado

- Buscar un problema de lógica y plantearlo como búsqueda en un espacio de estados.
- Indicar para dicho problema el estado inicial, el conjunto de posibles estados, el conjunto de operadores, el objetivo y el coste de los operadores.
- De los posibles métodos de búsqueda en el árbol de estados, seleccionar tres de ellos, en el que al menos uno se un recorrido guiado por información, y dibujar paso a paso cómo se van generando los nodos. Si el árbol es muy grande, dibujar los primeros niveles y dejar indicado cómo se van expandiendo los siguientes niveles. Para los recorridos no guiados por información, suponer que la preferencia es de arriba abajo y de izquierda a derecha.
- Indicar una heurística admisible para estimar el coste de un estado cualquiera hasta el estado meta.

Procedimiento de entrega de la actividad

- Subir mediante el Campus Virtual, un archivo Word o PDF con la solución de la tarea (Fecha límite 25/01/2014 23:59pm)