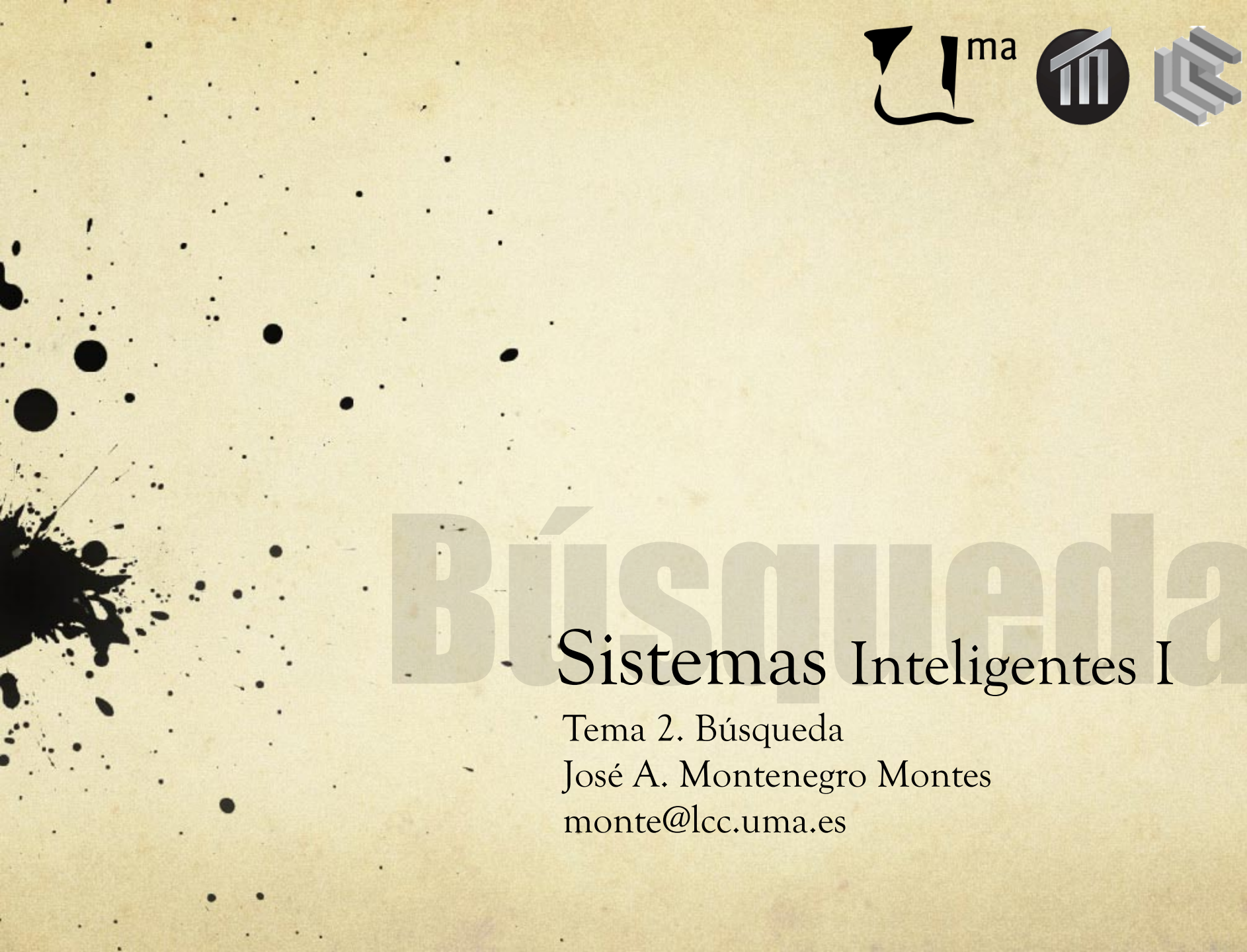


Sistemas Inteligentes I

Tema 2. Búsqueda

José A. Montenegro Montes
monte@lcc.uma.es

Resumen

- Agentes resolventes-problemas
- Búsqueda de Soluciones
- Estrategias Búsqueda
  - No informadas
  - Informadas
- Búsqueda Local
  - Gradiente
  - Recorrido Simulado
  - Algoritmo genético

Agentes resolventes-problemas

- Agentes inteligentes deben **maximizar** su rendimiento.
  - Elegir objetivo y satisfacerlo, implica **formulación objetivo**
  - **Formulación problema** , dado un objetivo, es el proceso de decidir acciones y estados a considerar.

- Agente con distintas opciones inmediatas de valores desconocidos puede decidir qué hacer, examinando las diferentes secuencias posibles de acciones que le conduzcan a estados de valores conocidos y entonces escoger la mejor secuencia.

- **Búsqueda** es el proceso de hallar la secuencia.
  - **Algoritmo de búsqueda** toma como entrada un problema y devuelve una solución (secuencia de acciones).

Definición Problema

- Problema puede definirse con 4 componentes:
  - **Estado inicial**, donde comienza agente.
  - **Acciones** disponibles por el agente, p.ej. Función sucesor.
  - **Test objetivo**, determina si un estado es objetivo.
  - **Costo camino**, asigna un costo numérico a cada camino.

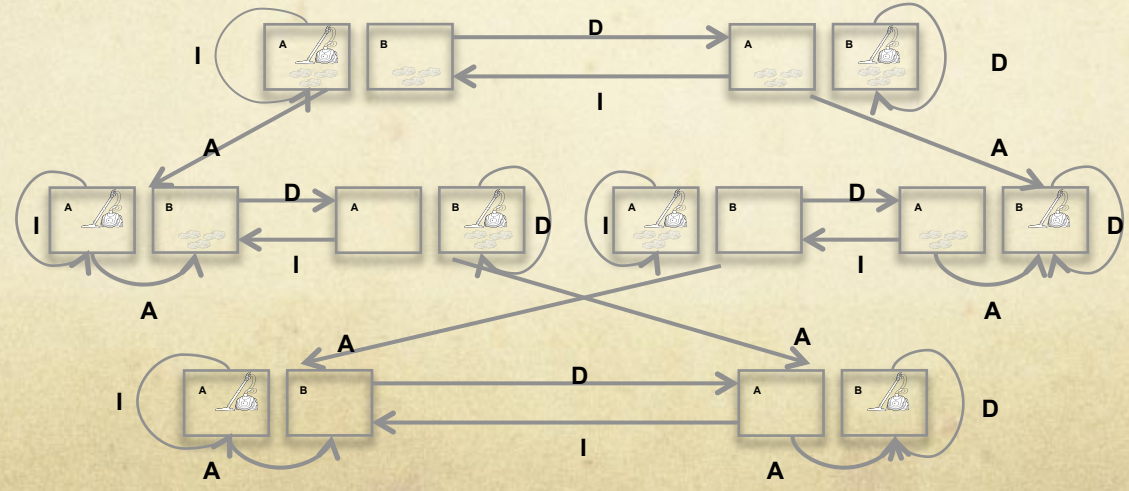
- **Espacio de estados**: Todos los estados alcanzables desde el estado inicial. Definido por el estado inicial y la función sucesor.
  - **Camino**: Secuencia estados conectados por una secuencia de acciones.

- **Solución**: Camino estado inicial a un estado objetivo.
  - Calidad solución viene dada por el costo del camino.
  - Solución Óptima coste mínimo de todas las soluciones.

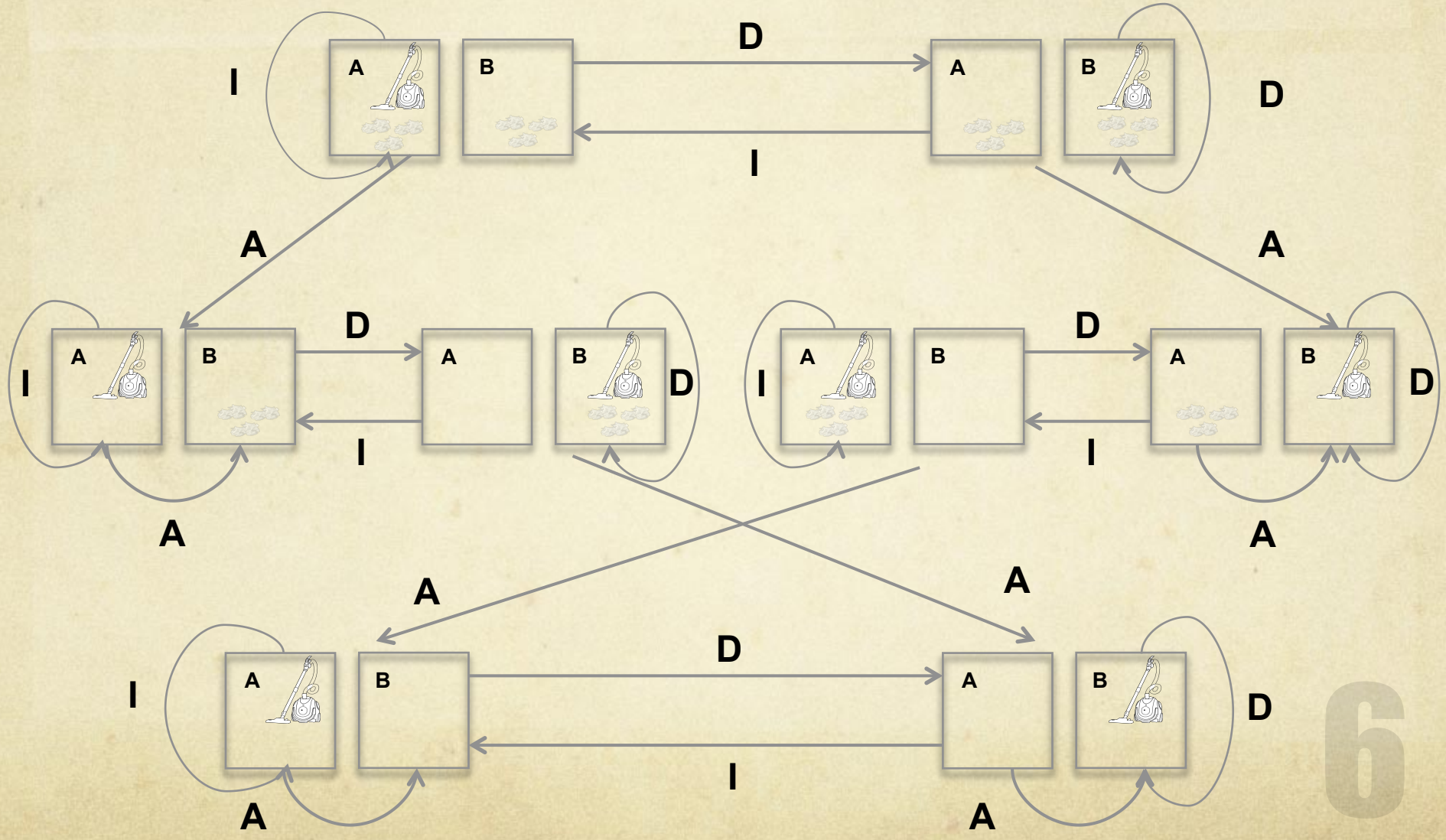
Ejemplo Problema

Mundo Aspiradora

- **Estados**: Dos localizaciones posibles y cada una puede estar sucia o no → 2x 2^2=8
- **Estado inicial:** Cualquier estado.
- **Función sucesor**: Tres acciones posibles (Izquierda, Derecha, Aspirar)
- **Test objetivo**, todos los estados cuadrados limpios.
- **Costo camino**, costo individual es 1, costo camino número pasos que lo componen.



Ejemplo Problema



Otro Ejemplo

8-puzzle

- **Estados**: Localización de las 8 fichas y el blanco.
- **Estado inicial:** Cualquier estado.
- **Función sucesor**: Cuatro acciones posibles (Izquierda, Derecha, Arriba, Abajo)
- **Test objetivo**, estados coincide con la configuración objetivo.
- **Costo camino**, costo individual es 1, costo camino número pasos que lo componen.

7	2	4
5		6
8	3	1

Posible estado inicial

	1	2
3	4	5
6	7	8

Estado final

Búsqueda Soluciones

Resolución problema

- Normalmente el espacio de estados generado por el estado inicial y la función sucesor es representado por un **árbol de búsqueda**.

- Cuando un estado es alcanzable por varios caminos tendremos un **grafo de búsqueda**.

- El **algoritmo general** sería comenzando en la raíz, escoge, comprueba y expande hasta que encuentra una solución o no existen más estados para expandir.
  - El estado ha expandir está determinado por la estrategia de búsqueda.

- **Frontera**: La colección (conjunto) de nodos (nodos hoja) que se generan pero no se ha expandido.
  - La estrategia de búsqueda será una función que seleccione del conjunto el siguiente nodo a expandir.

Rendimiento de la resolución

- **Completitud**: Garantizado algoritmo encuentra solución cuando existe.
- **Optimización**: Encuentra la estrategia la solución óptima.
- **Complejidad en tiempo**: Cuanto tarda en encontrar la solución.
- **Complejidad en espacio**: Cuanta memoria necesita para el funcionamiento de la búsqueda.

- **Eficiencia**:
  - **Coste de la búsqueda**, complejidad en tiempo y también uso memoria.
  - **Coste total,** coste de la búsqueda y coste del camino solución encontrado.

Estrategias Búsqueda

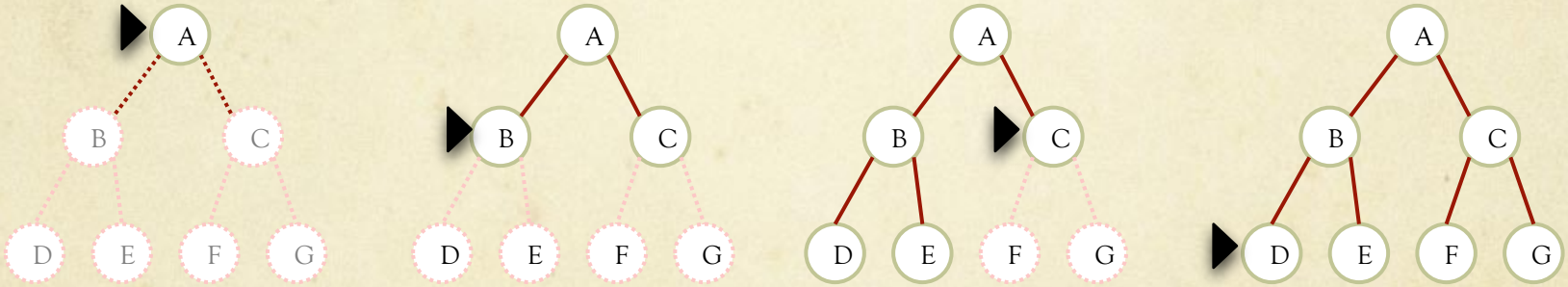
No informadas

Búsquedas no informadas

- **No** tienen **información adicional** acerca de los estados más allá de la que proporciona el problema.
  - Generan sucesores y distinguir entre un estado es objetivo o no.
- Estrategias:
  - Anchura
  - Anchura con costes (Dijkstra)
  - Profundidad
  - Profundidad limita
  - Profundidad progresiva
- Los problemas de búsqueda de **complejidad-exponencial** **no** pueden resolverse por métodos sin información, salvo casos pequeños.

Anchura

- Expanden todos los nodos a una **profundidad** en el árbol de búsqueda antes de expandir cualquier nodo del **próximo nivel**.
  - **Óptima** si todas las acciones tienen mismo coste (coste camino es una función no decreciente de la profundidad del nodo)



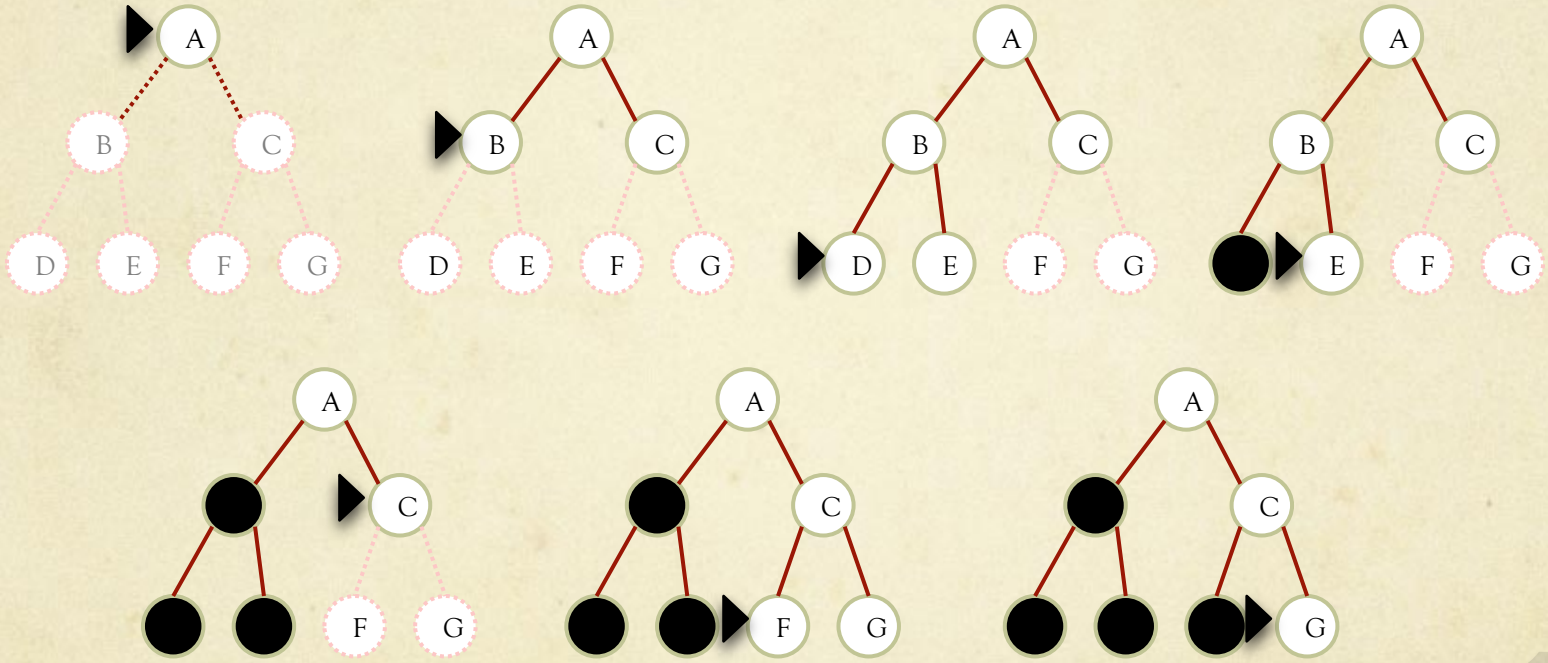
  - Tiene grandes requisitos de memoria más que tiempo de ejecución.

Anchura con costes

- Anchura es óptima cuando todos los costes son iguales.
  - Modificamos, de forma que sea óptimo con cualquier función coste.
- Expandimos el nodo n con el coste del camino más pequeño.
  - Si son todos iguales es idéntico a la búsqueda en anchura.
- No se preocupa por el número de pasos que tiene un camino, pero sí por su **coste total**.
  - Caso de bucle infinito si expande un nodo que tiene una acción de coste cero que conduce al mismo estado.

Profundidad

- Expande el nodo más profundo en la frontera actual del árbol de búsqueda.

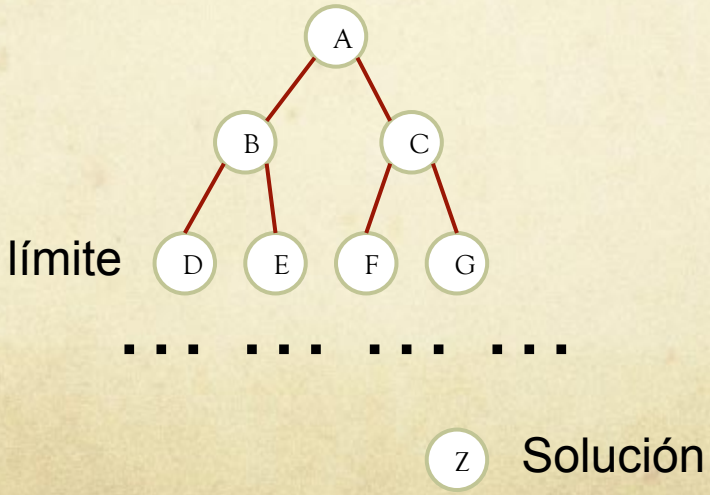


- Algoritmo no óptimo y no completo, ya que puede existir profundidad ilimitada.

Profundidad Limitada

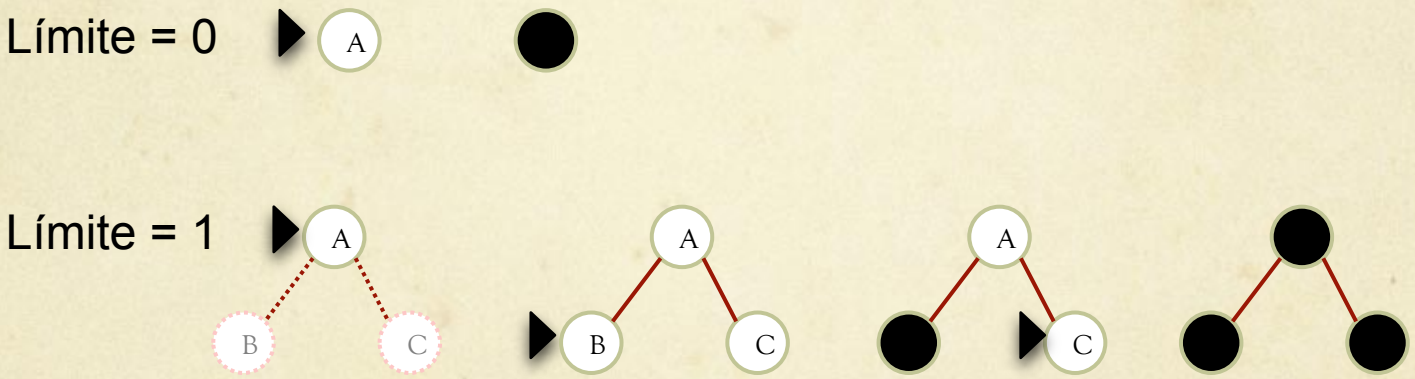
- Evitamos el problema de árboles ilimitados, estableciendo un límite de profundidad *l* predeterminado.
  - Nodos a profundidad *l* se les trata como si no tuviesen sucesor.

- Incompletitud si $l < d$, objetivo fuera alcance del límite profundidad.



Profundidad Progresiva

- Método de búsqueda no informada preferido cuando hay un espacio grande de búsqueda y no se conoce la profundidad de la solución.



Estrategias Búsqueda

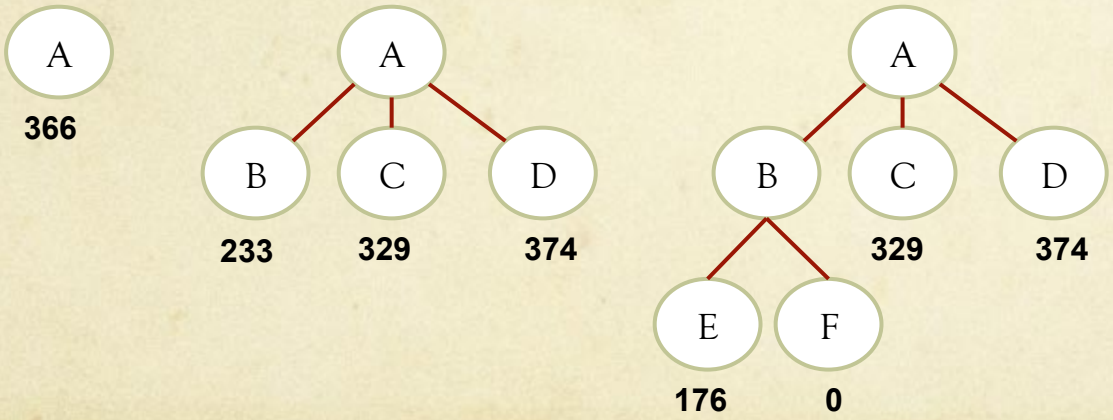
Informadas

Búsquedas informadas

- Estrategias de búsqueda informada puede encontrar soluciones de manera más eficiente que una estrategia no informada.
- Selecciona un nodo para la expansión basada en una función de evaluación f(n).
  - Normalmente se expande el nodo con la evaluación más baja.
- Función heurística h(n), forma más común de transmitir el conocimiento adicional del problema al algoritmo de búsqueda.

Algoritmo voraz (greedy)

- Expandir nodo más cercano al objetivo, basándose que conduce rápidamente a una solución.
  - *f(n)=h(n)*: coste estimado del camino más barato desde el nodo *n* a un nodo objetivo.
- Ejemplo h = Distancia en línea recta dos ciudades.

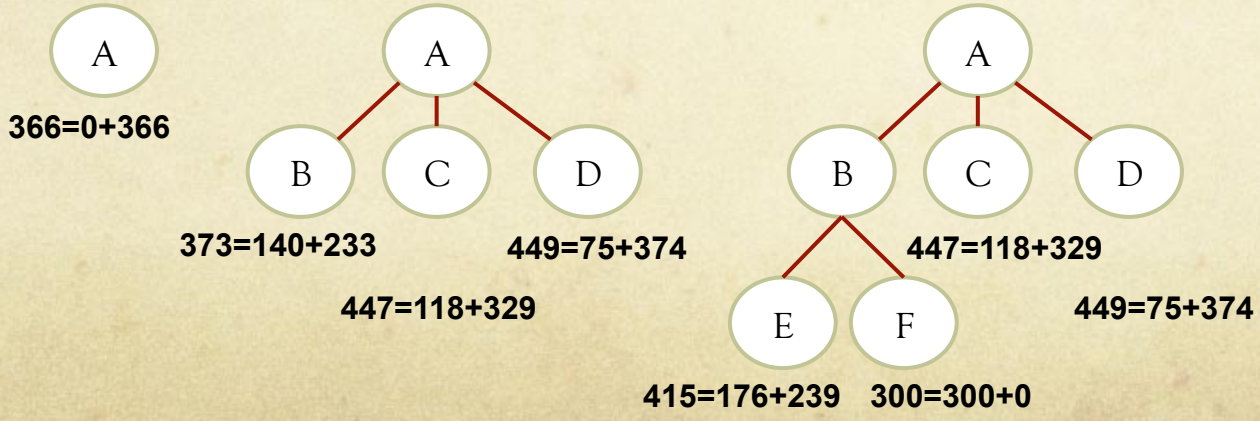


Ciudad	Distancia F
A	366
B	233
C	329
D	374
E	176
F	0

Algoritmo no óptimo e incompleto

Búsqueda A*

- Evalúa los nodos combinando g(n), el coste para alcanzar el nodo, y h(n), el coste de ir al nodo objetivo:
  - $f(n)=g(n)+h(n)$
- A* es óptima si h(n) es una heurística admisible, h(n) nunca sobreestime el coste de alcanzar el objetivo.
  - Funciones optimistas, el coste resolver el problemas es menor que en la realidad.

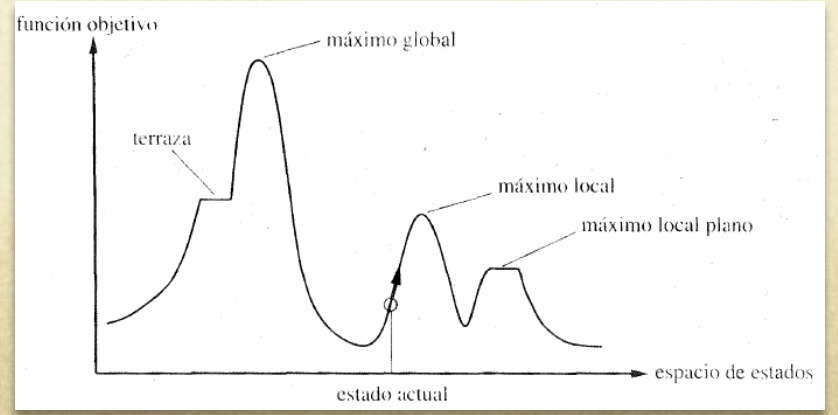


Ciudad	Distancia F
A	366
B	233
C	329
D	374
E	176
F	0

Búsqueda Local

Búsqueda Local

- Algoritmo anteriores explorar sistemáticamente espacios de búsqueda. Camino al objetivo es parte de la solución del problema.
  - Existen problemas el camino al objetivo es irrelevante.
- Son útiles para resolver problemas de optimización, encontrar el mejor estado según función objetivo.
- Espacio de búsqueda formado por estados
  - Cada estado es una posición
  - La altura es el valor de la función objetivo



Gradiente

- Bucle que se mueve continuamente en dirección del valor creciente, es decir, cuesta arriba.
- Finaliza cuando alcanza "un pico" donde ningún vecino tiene un valor más alto.
- El algoritmo no mantiene un árbol de búsqueda, solo nodo actual y valor función objetivo.

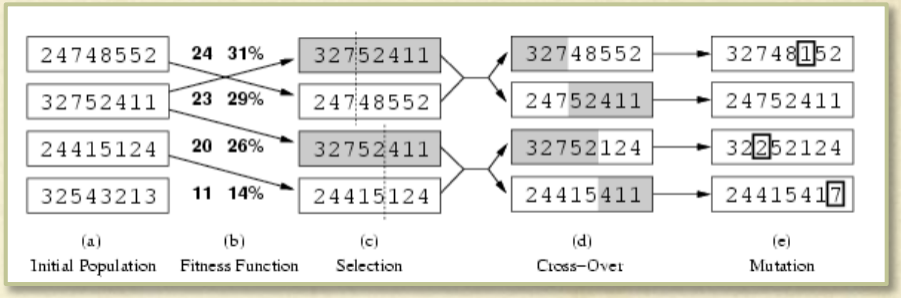
```
function HILL-CLIMBING( problem) returns a solution state
        inputs: problem, a problem
        static: current, a node
                next, a node
current← MAKE-NODE(INITIAL-STATE[problem])

loop do
next← a highest-valued successor of current
if VALUE[next] < VALUE[current] then return current
current←next
```

Recorrido Simulado

- Algoritmo gradiente no realiza movimientos “cuesta abajo”, no es completo ya que puede estancarse en un máximo local.
- Un algoritmo aleatorio, es completo, pero puramente ineficaz.
- Combinamos las dos visiones que permita un algoritmo eficaz y completo.
- Comienza a una temperatura alta y luego reduciendo gradualmente la intensidad, a una temperatura más baja.
- Algoritmo parecido anterior, en vez de escoger el mejor movimiento, escoge un movimiento aleatorio.

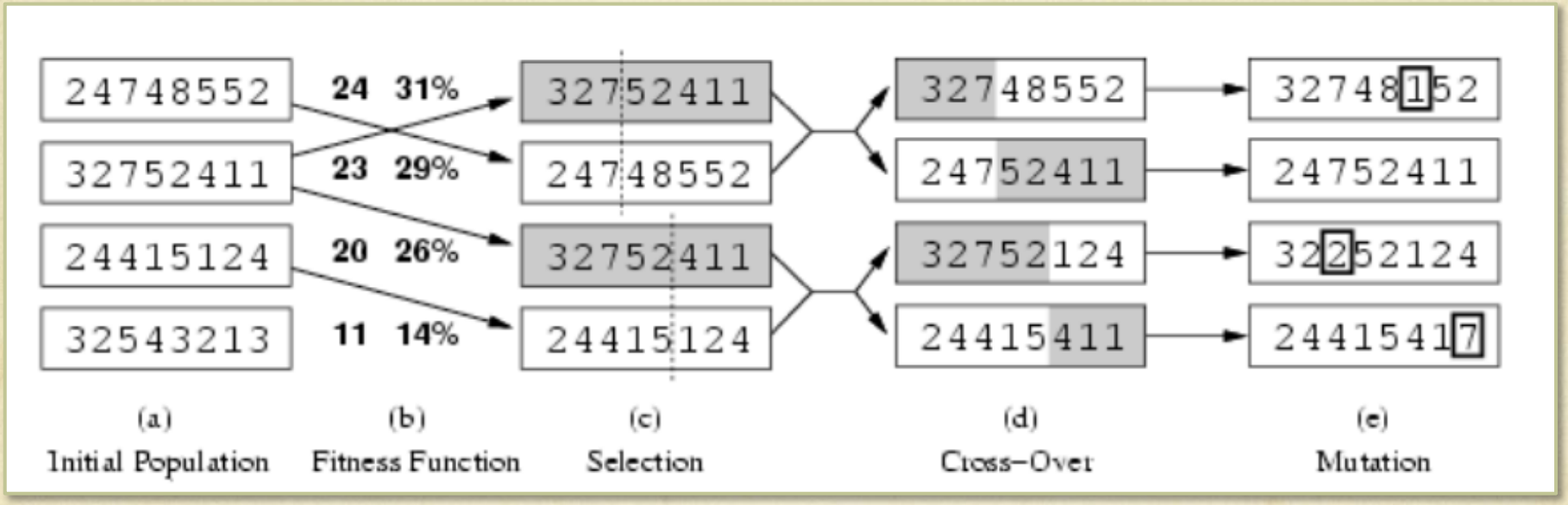
Algoritmo genéticos

- Los sucesores son generados combinando dos estados padres, más que modificando un solo estado.



- Definiciones:
  - **Población:** Comenzamos con un conjunto de *k* estados generados de forma aleatoria.
  - **Individuo**: Representado cadena sobre alfabeto finito.
  - **Función idoneidad**: Cada estado se tasa con una función de evaluación. Valores más altos para los mejores valores.
  - **Cruce**: Seleccionamos dos pares de forma aleatoria para la reproducción, según con las probabilidades.
    - Además se selecciona aleatoriamente un punto de cruce.
  - **Mutación**: Finalmente se realiza una mutación aleatoria con una probabilidad independiente.

Algoritmo genéticos



Ejercicios

Ejercicio 1

- Las distancias por carretera entre las ciudades de una región vienen dadas en kilómetros en la tabla siguiente:

	A	B	C	D	E	F	G
A		10	15				
B				5	15		
C					5	15	
D							20
E						5	5
F							20
G							

- La siguiente tabla contiene una estimación optimista de la distancia en kilómetros desde la ciudad G a cada una de las demás ciudades:

	A	B	C	D	E	F	G
Estimación	19	9	9	4	4	3	0

- Aplicar el algoritmo A* para encontrar un camino desde la ciudad A a la ciudad G. Indicar **para cada iteración** el nodo seleccionado para expansión, sus sucesores y los valores correspondientes de g(n), h(n) y f(n). Mostrar el árbol de búsqueda resultante, detallando las operaciones realizadas sobre el mismo.

Ejercicio d1

- Propón una representación adecuada de los estados.
- Describe el espacio de búsqueda.
- Detalla las acciones disponibles y sus costes.
- ¿Cuál es la condición que debe cumplir un estado para ser final?
- ¿Cuántos estados finales hay?

Ejercicio d1.a

- Una persona sale de Gerona y otra de Cádiz.
- En cada etapa, ambas personas pueden moverse desde la capital de una provincia peninsular de España a la capital de una provincia vecina.
- **La tarea es encontrar una ruta** para cada persona de manera que se encuentren en alguna capital lo antes posible.
  - El viaje hacia la siguiente capital lo empiezan ambas personas simultáneamente, y ambas deben llegar a sus destinos antes de empezar con la siguiente etapa.
  - Suponemos que sabemos el tiempo necesario para viajar desde cualquier capital a cualquier otra capital vecina.

Ejercicio d1.a

Propón una representación adecuada de los estados.

Un estado es un par ordenado (*a*,*b*), donde *a* es la posición actual de la primera persona, y *b* es la posición actual de la segunda persona. Ambas posiciones deben ser capitales de provincias peninsulares españolas.

Describe el espacio de búsqueda.

El espacio de estados es el conjunto de todos los pares ordenados con la estructura anterior. Como hay 47 provincias peninsulares españolas → 47^2.

Detalla las acciones disponibles y sus costes.

Mover (a,b) (c,d). Coste máx(*distancia*(*a*,*c*), *distancia*(*b*,*d*)), dado que la persona que llega antes a su destino debe esperar a la otra.

¿Cuál es la condición que debe cumplir un estado para ser final?

Un estado es final si *a*=*b*, es decir, si ambas personas están en el mismo sitio.

¿Cuántos estados finales hay?

Hay 47 estados finales de la forma (*a*,*a*), donde *a* es provincia

Ejercicio d1.b

- Un grupo de 5 personas quiere cruzar un viejo y estrecho puente por la noche. Es necesario usar una linterna para cruzarlo.
- Sólo hay una linterna disponible, cuya batería está baja: sólo quedan 5 minutos antes de que se apague.
- Las personas necesitan 10, 30, 60, 80 y 120 segundos para cruzar el puente, respectivamente.
- El puente sólo puede sostener a 2 personas como mucho al mismo tiempo, y la velocidad a la que se cruza es la de la persona más lenta.
- La linterna no se puede lanzar de un lado al otro, así que es necesario que una persona cruce el puente de nuevo para llevarla al otro lado.
- La tarea es encontrar el modo más rápido de que todo el mundo llegue al lado contrario, si esto es posible.

Ejercicio d1.b

Propón una representación adecuada de los estados.

Un estado es una terna (A, B, c), donde A es el conjunto de personas en el lado inicial del puente, B es el conjunto de personas en el lado final del puente, y c es la ubicación (inicial/final) de la linterna. Notaremos a cada persona con el número de segundos que necesita para cruzar el puente.

Describe el espacio de búsqueda.

El espacio de estados es el conjunto de todas las ternas con la estructura anteriormente citada, donde los dos conjuntos forman una partición del conjunto {10, 30, 60, 80, 120}, y $c \in$ {inicial, final}.

Detalla las acciones disponibles y sus costes.

Tiempo agotado Si el coste de camino g(n) desde el estado inicial ({10, 30, 60, 80, 120}, ∅, inicial) al estado actual n es mayor o igual que 300 segundos.

El coste de una acción es el mayor de los valores de las personas que han cruzado.

¿Cuál es la condición que debe cumplir un estado para ser final?

Para saber si un estado es final comprobamos si B tiene 5 elementos, es decir, si todas las personas están en el lado final del puente.

¿Cuántos estados finales hay?

Sólo hay un estado final: (∅, {10, 30, 60, 80, 120}, final). Nótese que no hay manera de que la linterna acabe en el lado inicial del puente si todo el mundo está en el lado final.

Ejercicio d1.c

- Una empresa de panadería tiene una furgoneta para realizar el reparto diario de pan, que puede cargar hasta P panes.
- Debe servir a N restaurantes, y el i-ésimo restaurante necesita c_i panes diariamente.
- La cantidad total de panes que repartir excede a P, pero hay panaderías de la empresa repartidas por toda la ciudad, donde la furgoneta puede recargar si es necesario.
- Supondremos que la furgoneta empieza el recorrido en alguna de las panaderías.
- La tarea es encontrar la ruta más corta para servir a todos los restaurantes, incluyendo las visitas a las panaderías que sean necesarias.
  - Supondremos conocidas las distancias entre cualquier par de puntos de la ciudad.

Sistemas Inteligentes

José A. Montenegro Montes

monte@lcc.uma.es