Razonamiento y planificación automática

Nerea Luis Mingueza / Alejandro Cervantes

Tema 5: Búsqueda informada

Índice de la clase

Tema 4

Continuación de búsqueda no informada

Tema 5

Búsqueda informada

¿Qué es una heurística?

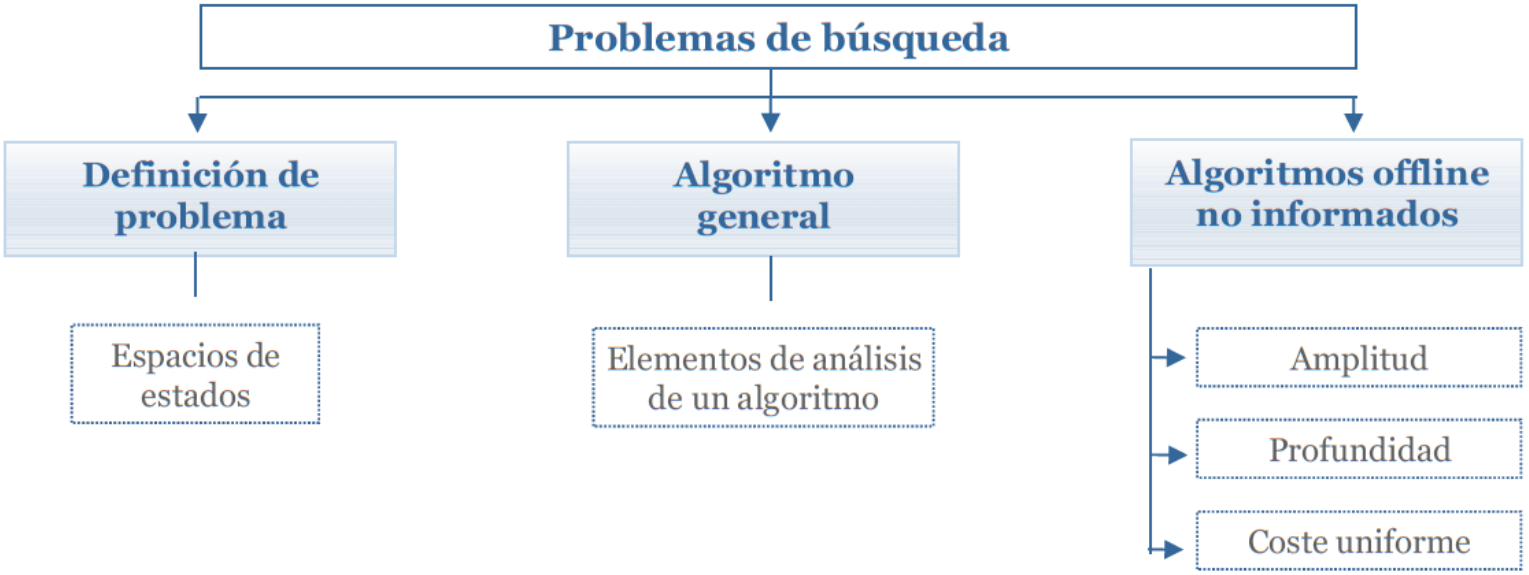
El algoritmo de búsqueda A* (intro)

Actividad 1

Presentación de la actividad 1

Continuación Tema 4

En la clase pasada...



Búsqueda

MÉTODO

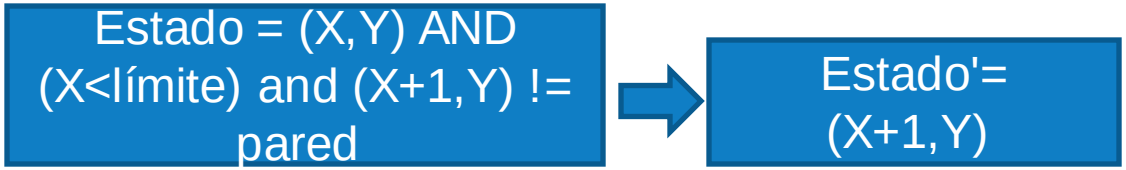
1. Se representan los estados del problema
2. Se especifican y representan las acciones (operadores)
3. Se especifica cómo detectar el estado objetivo (meta)
4. Se escoge un algoritmo (Amplitud, Profundidad, etc.) y se ejecuta

Ejemplo: laberinto (sólo mover)

1. Estados: (X,Y) Ejemplo: X horizontal e Y vertical empezando desde arriba

2. Acciones:

Mover derecha:



Mover izquierda, arriba y abajo (similares). Según casos podríamos tener otros movimientos (ej: ajedrez)

3. Objetivo (goal):

Estado = Puerta de salida

https://qiao.github.io/PathFinding.js/visual/

Búsqueda no informada

ALGORITMO GENERAL: sacar el primer nodo de la lista abierta; si es meta, termina; si no, expandir (generar TODOS[1] sus sucesores), e insertar en la lista abierta.

[1]NOTA: Algunos nodos repetidos no se generan

AMPLITUD: los nuevos nodos se introducen al final de ABIERTA (cola, FIFO)

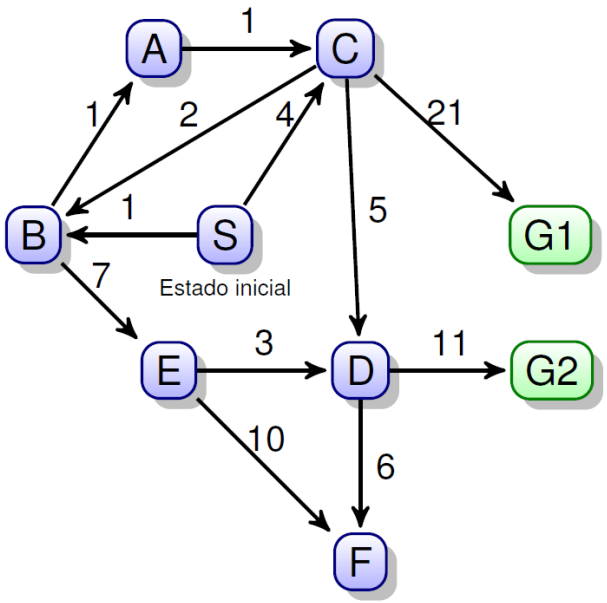
PROFUNDIDAD: los nuevos nodos se introducen al principio de ABIERTA (pila, LIFO)

UCS: la lista ABIERTA se ordena por COSTE TOTAL (desde el inicio hasta el nodo) (cola con prioridad)

Los algoritmos no informados sí tienen información completa sobre la representación del dominio del problema pero no sobre la manera más eficaz de resolver los problemas de este dominio

Ejercicio Búsqueda en Amplitud

S: estado inicial, G1 y G2 metas, ignorar los costes en los arcos.
Al expandir, el orden es alfabético



Al generar nodos seguimos el orden alfabético indicado

Se añade a la lista abierta por detrás (los más antiguos se escogen antes)

Expandido	Genera (Sucesores)	Abierta (va acumulando)
S	B, C	B, C
B	A, E	C, A, E
C	D, G1 (anotamos que llegamos a G1 desde C)	A, E, D, G1
A	(C no se genera por repetido)	E, D, G1
E	F (D no se genera por repetido)	D, G1, F
D	G2 (F no se genera por repetido)	G1, F, G2
** G1	FIN	

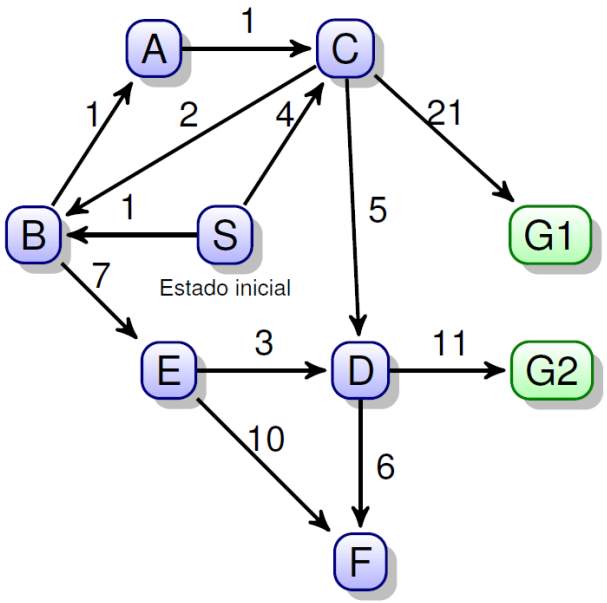
1 Iteraciones 7

No terminamos hasta que no corresponde expandir un nodo objetivo (aunque lo tengamos en la lista Abierta). No generamos ningún nodo con un estado repetido (que esté o haya estado en Abierta)

Solución: *SC,CG1 .* Tenemos que retrazar a mano el camino porque no pintamos los arcos en la tabla, en el código se irían guardando. Se llega al objetivo **G1**.
Longitud solución: 2**, Expandidos**: 6 (columna 1), **Generados**: 8 (columna 2)

Ejercicio Búsqueda en Profundidad

S: estado inicial, G1 y G2 metas, ignorar los costes en los arcos.
Al expandir, asumamos orden inverso al alfabético



Al generar nodos seguimos el orden indicado

Se añade a la lista abierta por delante (los más recientes se escogen antes)

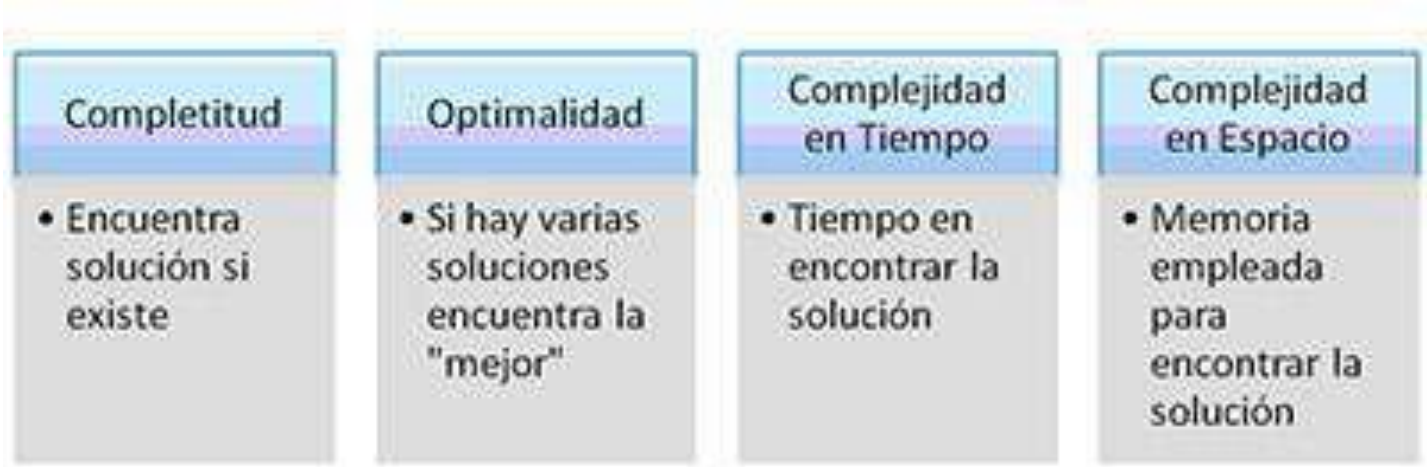
Expandido	Genera (Sucesores)	Abierta (va acumulando)
S	C, B	**B**, **C**
B	E, A	**A**, **E** , C
A	(C no se genera por repetido)	E, C
E	F, D	**D**, **F**, C
D	G2 (F no se genera por repetido)	**G2**, F, C
** G2	FIN	

1 Iteraciones 6

No terminamos hasta que no corresponde expandir un nodo objetivo (aunque lo tengamos en la lista Abierta). No generamos ningún nodo con un estado repetido (que esté o haya estado en Abierta)

Solución: *SB, BE, ED, DG2 .* Tenemos que retrazar a mano el camino porque no pintamos los arcos en la tabla, en el código se irían guardando. Se llega al objetivo **G2**.
Longitud solución: 4**, Expandidos**: 5 (columna 1), **Generados**: 7 (columna 2)

Características de los algoritmos



COMPLETITUD:
Al introducir el mecanismo de lista abierta, todos los algoritmos que mostramos van a ser completos (a costa de mayor complejidad espacial, es decir, memoria).
Ver Russell&Norvig para algoritmos no completos (profundidad pura)

OPTIMALIDAD:
Sin costes: los algoritmos óptimos encuentran un solución de mínima profundidad
Con costes: los algoritmos óptimos encuentran la solución de mínimo coste

Búsqueda con costes

Búsqueda con coste uniforme (UCS/Dijkstra)

Usa el Algoritmo Genérico con las siguientes peculiaridades:

- En cada nodo anotamos la suma de los costes de las acciones que me han llevado al mismo g(N)
- Los nuevos sucesores se añaden ordenados en función del coste total hasta el nodo (g(N)).
- Se insertan en la lista Abierta[1] nodos si y solo si a) su estado no esté en ella; o b) estando ya en la lista, el coste G del nodo es menor (y reemplaza al anterior).
- La lista abierta funcionará como cola de prioridad ordenada según G.
- Adicionalmente, evita ciclos generales (no escribe nodos que se han generado previamente en el camino al nodo que se expande).
- Termina en cuanto toca expandir un nodo meta

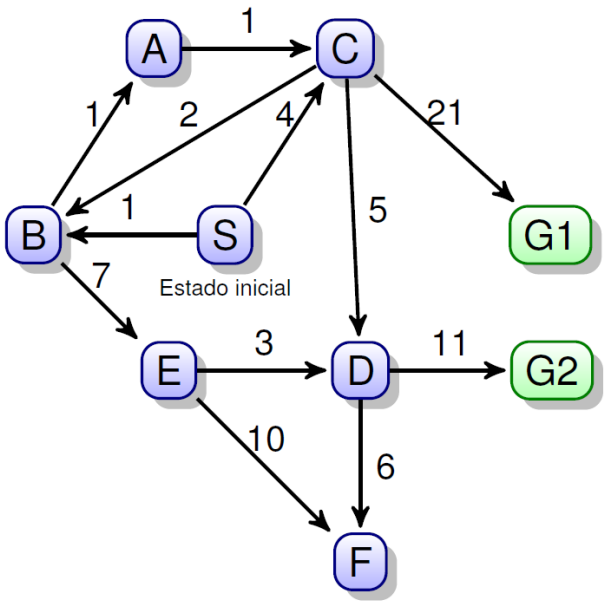
[1]Opcionalmente, usa lista cerrada para evitar repeticiones innecesarias: si un nodo estuvo en la lista abierta, y se expandió (desapareciendo), se guarda aquí. Se evita también introducir en Abierta nodos con el mismo estado, pero peor coste, que los que están en Cerrada.

Óptimo en coste

Complejidad exponencial en espacio y tiempo

Búsqueda con coste uniforme (UCS)

S: estado inicial, G1 y G2 metas, Costes en los arcos. Al expandir, el orden es alfabético, y también se usa este orden para empates en Abierta (a igualdad de g)



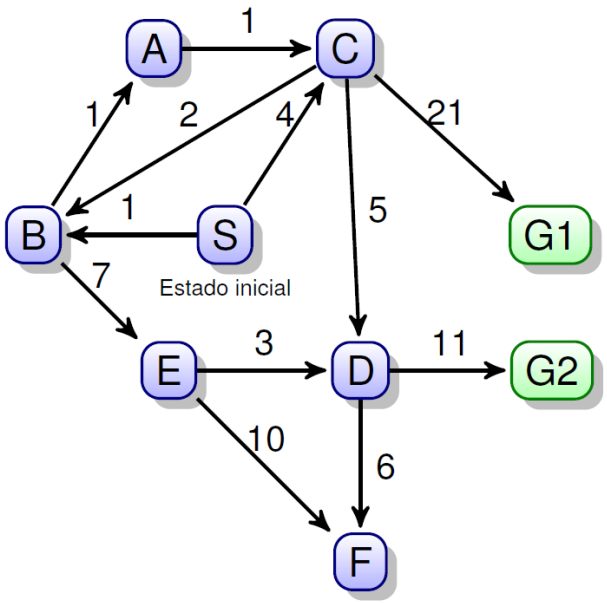
Expandido	Genera	Abierta	Cerrada

1

8

Búsqueda con coste uniforme (UCS)

S: estado inicial, G1 y G2 metas, Costes en los arcos. Al expandir, el orden es alfabético, y también se usa este orden para empates en Abierta (a igualdad de g)



Expandido	Genera	Abierta	Cerrada
S	B (g=1), C(g=4)	**B (g=1)**, **C(g=4)**	**S**
B (g=1)	A (g=1+1=2), E (g=1+7=8)	**A (g=2)**, C(g=4), **E(g=8)**	**S,** B (g=1)
A (g=2)	C (g=2+1=3)	**C(g=3)**, ~~C(g=4),~~ E(g=8)	**S,** B (g=1), A(g=2)
C (g=3)	D (g=3+5=8), G1 (g=3+21=24) (B repetido en el camino a C)	**D(g=8)**, E(g=8), **G1 (g=24)**	**S,** B (g=1), A(g=2), C(g=3)
D (g=8)	F (g=8+6=14), G2 (g=8+11=19)	E(g=8), **F(g=14)**, **G2 (g=19)**, G1 (g=24)	**S,** B (g=1), A(g=2), C(g=3), D(g=8)
E (g=8)	D (g=8+3=11) (peor que el de la lista cerrada), F (8+10=18) (peor)	F(g=14), G2 (g=19), G1 (g=24)	**S,** B (g=1), A(g=2), C(g=3), D(g=8),E(g=8)
F (g=14)	No tiene sucesores	G2 (g=19), G1 (g=24)	**S,** B (g=1), A(g=2), C(g=3), D(g=8),E(g=8),F(g=14)
** G2 (g=19)	FIN		

1

8

Solución: *SB(1),BA(1),AC(1),CD(5),DG2(11)* . Tenemos que retrazar a mano el camino de coste 19 porque no pintamos los arcos en la tabla, en el código se irían guardando.
Coste: 1+1+1+5+11=19, **Expandidos**: 7, **Generados**: 11 (aunque algunos nunca se insertaron en la lista abierta)

Ejemplos online de búsqueda no informada

https://tristanpenman.com/demos/n-puzzle/

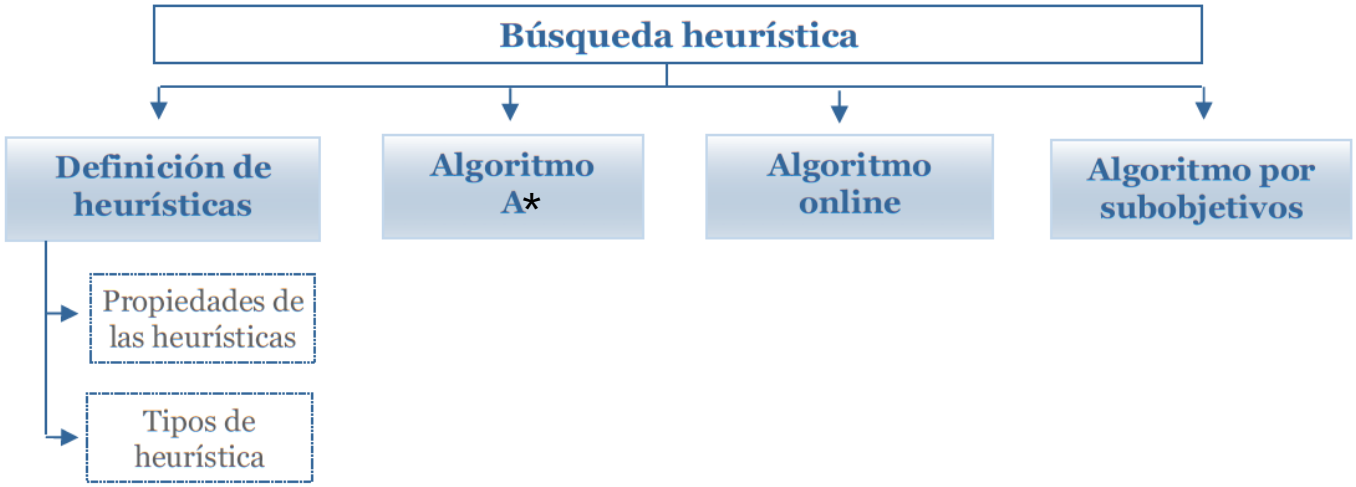
https://workshape.github.io/visual-graph-algorithms/#bfs-visualisation

https://hurna.io/academy/algorithms/maze_pathfinder/bfs.html

https://graphonline.ru/en/

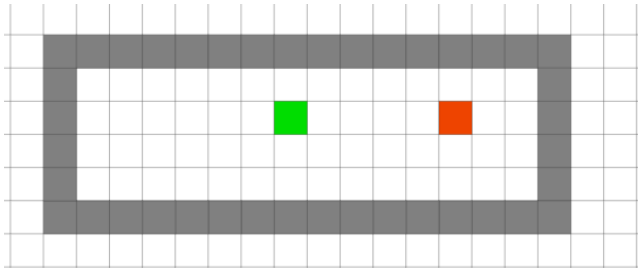
https://algorithms.discrete.ma.tum.de/graph-algorithms/spp-dijkstra/index_en.html

Búsqueda informada

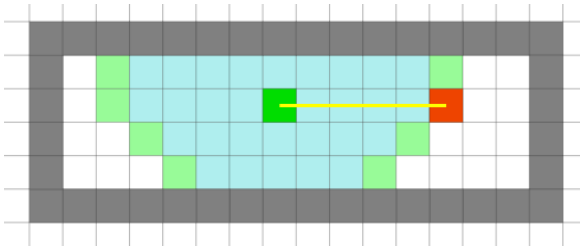

Búsqueda heurística
Definición de heurísticas
Algoritmo A*
Algoritmo online
Algoritmo por subobjetivos
Propiedades de las heurísticas
Tipos de heurística

Motivación

¿Podemos diseñar un mecanismo riguroso para que la búsqueda evite expandir estados que no ayudan a resolver el problema?

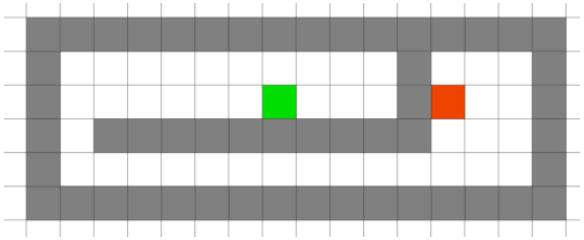


Verde: estado inicial; rojo, estado final; movimientos N,S,E y W



Amplitud: verde, nodos en lista ABIERTA. Azul, nodos expandidos. Amarillo, camino encontrado (solución)

Una buena regla sería: siempre prefiero mover a un estado más "cercano" a la meta ("cercanía" = información extra sobre el dominio "laberintos con movimiento ortogonal)



… pero no a cosa de "atascarse" en la pared... a veces hay que retroceder para poder avanzar más tarde

www.unir.net