Búsqueda Informada

La búsqueda informada es la que utiliza el conocimiento específico del problema más allá de la definición del problema en sí mismo. Esta búsqueda informada puede encontrar soluciones de una manera más eficiente que una estrategia no informada. Se selecciona un nodo para la expansión basada en una **función de evaluación f(n).**

Un componente clave de estos algoritmos es una **función heurística** denotada h(n)

h(n)= costo estimado del camino más barato desde el **nodo n** a un nodo objetivo

Si n es un nodo objetivo h(n) = 0

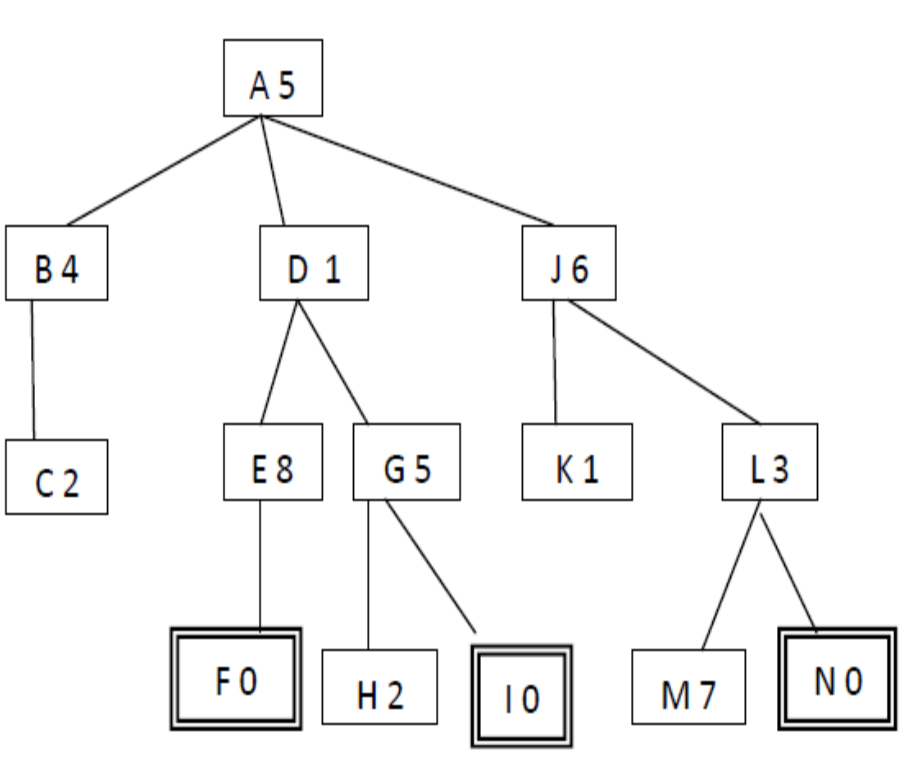
Búsqueda primero el mejor

Trata de expandir el nodo más cercano al objetivo, alegando que probablemente conduzca rápidamente a una solución. Así evalúa los nodos utilizando solamente la función heurística f(n) =h(n)

La búsqueda primero el mejor se parece a la búsqueda primero en profundidad en el modo que prefiere seguir un camino hacia el objetivo, pero volverá atrás cuando llegue a un callejón sin salida. Sufre los mismos defectos que la búsqueda primero en profundidad, **no es óptima y es incompleta**, porque puede ir hacia abajo en un camino infinito y nunca volver para intentar otras posibilidades. La complejidad en tiempo y espacio, del caso peor, es $O(b^m)$ donde **m es la profundidad máxima del espacio de búsqueda**. Con una buena función heurística se puede reducir la complejidad considerablemente. La cantidad de la reducción depende del problema particular y de la calidad de la heurística.

Búsqueda Informada Primero el mejor

Nodos meta F,I y N



Lista Abierta		Lista Cerrada
	A5	-
m=A	D1, B4, J6	A
m=D	B4, G5, J6, E 8	A, D
m= B	C2, G5, J6, E 8	A, D, B
m= C	G5, J6, E 8	A, D, B, C
m= G	I0, H2, J6, E 8	A,D, B, C, G
m= I		
Llega a I nodo meta		

Búsqueda Informada A*

Búsqueda A*: minimizar el costo estimado total de la solución
Evalúa los nodos combinando g(n) el costo para alcanzar el nodo y h(n) el costo de ir al nodo objetivo.

f(n)= g(n) + h(n)

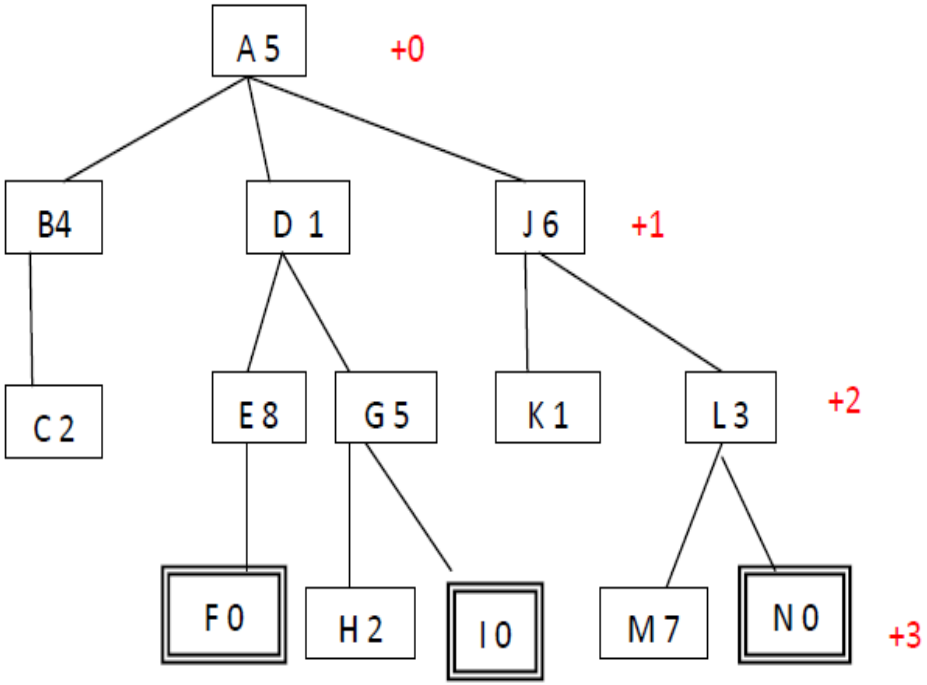
f(n) = costo más barato estimado de la solución a través de n

Si tratamos de encontrar la solución más barata es razonable intentar primero el nodo con el valor más bajo de g(n) + h(n), siempre y cuando la función heurística satisfaga ciertas condiciones la búsqueda A* es completa y óptima.
A* es óptima si h(n) es una heurística admisible, es decir, con tal que la h(n) nunca sobrestime el costo de alcanzar el objetivo.
La heurística debe ser también consistente, es decir si para cada **nodo n** y cada **sucesor n'** de n generado por cualquier **acción a**, el costo estimado de alcanzar el objetivo desde **n** no es mayor que el costo de alcanzar **n'** mas el costo estimado de alcanzar el objetivo desde n'.

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

Búsqueda Informada A*

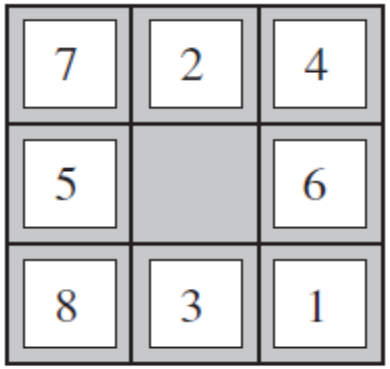
Nodos objetivos F, I, N



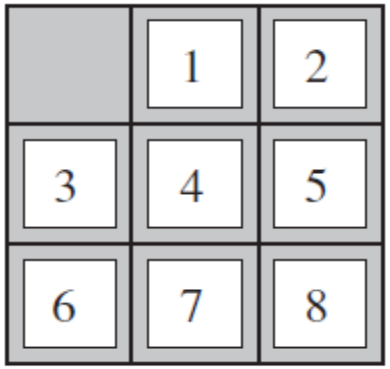
Lista Abierta		Lista Cerrada
	A	-
m=A	D2, B5, J7	A
m=D	B5, J7, G7, E10	A, D
m= B	C4, J7, G7, E10	A, D, B,
m=C	J7, G7, E10	A, D, B, C
m= J	K3, L5, G7, E10	A, D, B, C, J
m= K	L5, G7, E10	A, , D, B, C, J, K
m = L	N3, G7, E10, M10	A, D, B, C, J, K, L
m= N		
Llega a N nodo meta		

Funciones heurísticas

El 8-puzle fue uno de los primeros problemas de búsqueda heurística, el objeto del puzle es deslizar las fichas horizontalmente o verticalmente al espacio vacío hasta que la configuración empareje con la configuración objetivo.



Estado inicial



Estado objetivo

Cuando la ficha vacía está en el medio hay cuatro movimientos posibles, cuando está en una esquina hay dos y cuando está a lo largo del borde hay tres. En el caso del 8-puzle son aproximadamente 22 pasos con un factor de ramificación de 3, descartando los estados repetidos podríamos reducirlo a un factor de aproximadamente 170,000 estados posibles

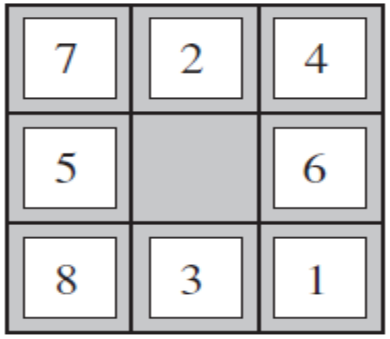
Funciones heurísticas

En el caso del 15-puzle, ya se complica más, las funciones heurísticas comúnmente usadas son:

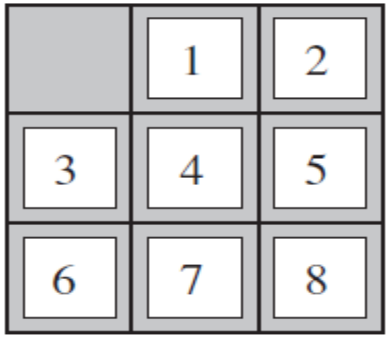
- h1= número de piezas mal colocadas, si todas las piezas estuvieran fuera de su posición el valor de h1= 8 para el 8-puzle.
- h2= suma de las distancias de las piezas a sus posiciones en el objetivo, la distancia que contaremos será las suma de las distancias horizontales y verticales. Las piezas 1 a 8 en el caso de que estuvieran todas fuera de su posición nos da una distancia de:

$$h2 = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$$

Ninguna de estas dos funciones heurísticas sobrestima el costo solución verdadero que es 26



Estado inicial



Estado objetivo

El efecto de la precisión heurística en el rendimiento

Una manera de caracterizar **la calidad** de una heurística es el b* factor de ramificación eficaz. Si el número **total de nodos generados** por A* para un problema particular es **N**, y **la profundidad de la solución es d**, **entonces b* es el factor de ramificación que un árbol uniforme de profundidad d debería tener para contener N + 1 nodos**. Así

$$N + 1 = 1 + b^* + (b^*)^2 + \ldots\ldots + (b^*)^d$$

El factor de ramificación eficaz puede variar según los ejemplos del problema, pero por lo general es constante para problemas suficientemente difíciles. Una heurística bien diseñada tendría un **valor de b* cerca de 1**, permitiría resolver problemas bastante grandes.
Se probaron **las heurísticas h1 y h2** y los resultados sugieren que A* con h2 es mas eficiente.
h1 y h2 son estimaciones de la longitud del camino restante para el 8-puzle, pero también son longitudes de caminos absolutamente exactos para versiones simplificadas del puzle. Si se cambiaran las reglas del puzle de modo que una ficha pudiera moverse a todas partes, entonces tanto **h1 como h2 darían el número exacto de pasos en la solución más corta**. A un problema con menos restricciones se le llama **problema relajado**. El costo de una **solución óptima** en un problema relajado es un **heurística admisible** para el problema original.

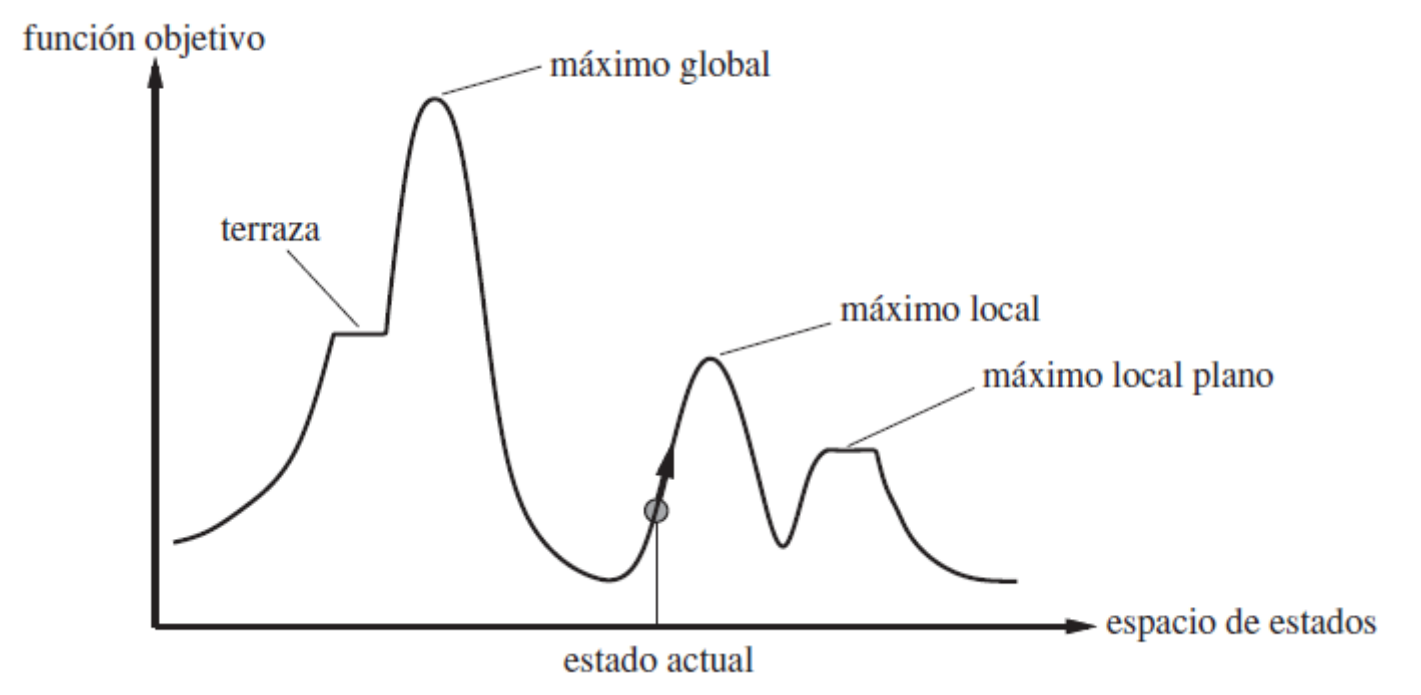
El efecto de la precisión heurística en el rendimiento

Si el problema relajado es difícil de resolver, entonces los valores de la correspondencia heurística serán costosos de obtener.

Algoritmos de búsqueda local y problemas de optimización

En muchos problemas, **el camino al objetivo es irrelevante**, por ejemplo el problema de las 8-reinas, lo que importa es la configuración final de las reinas, no el orden en las cuales se añaden. Esta clase de problemas incluyen muchas aplicaciones importantes como diseño de circuitos integrados, disposición del suelo de una fábrica, programación del trabajo en tiendas, programación automática, optimización de redes de telecomunicaciones, dirigir un vehículo, etc.
Si no importa el camino al objetivo, podemos considerar una clase diferente de algoritmos que no se preocupen de los caminos. **Los algoritmos de búsqueda local funcionan con un solo estado actual y generalmente se mueve sólo a los vecinos del estado.** Aunque los algoritmos de búsqueda local no son sistemáticos, tienen **dos ventajas importantes, primero usan poca memoria y segundo pueden encontrar a menudo soluciones razonables en espacios de estados grandes o infinitos** (continuos) para los cuales son inadecuados los algoritmos sistemáticos.

Algoritmos de búsqueda local y problemas de optimización



Un paisaje del espacio de estados unidimensional en el cual la elevación corresponde a la función objetivo. El objetivo es encontrar el máximo global.

Preguntas?