Razonamiento y Planificación Automática

Búsqueda heurística

Índice

Esquema 3

Ideas clave 4

6.1. ¿Cómo estudiar este tema? 4

6.2. Tipos de heurísticas 5

6.3. Búsqueda A* 9

6.4. Búsqueda por subobjetivos 13

6.5. Búsqueda online 16

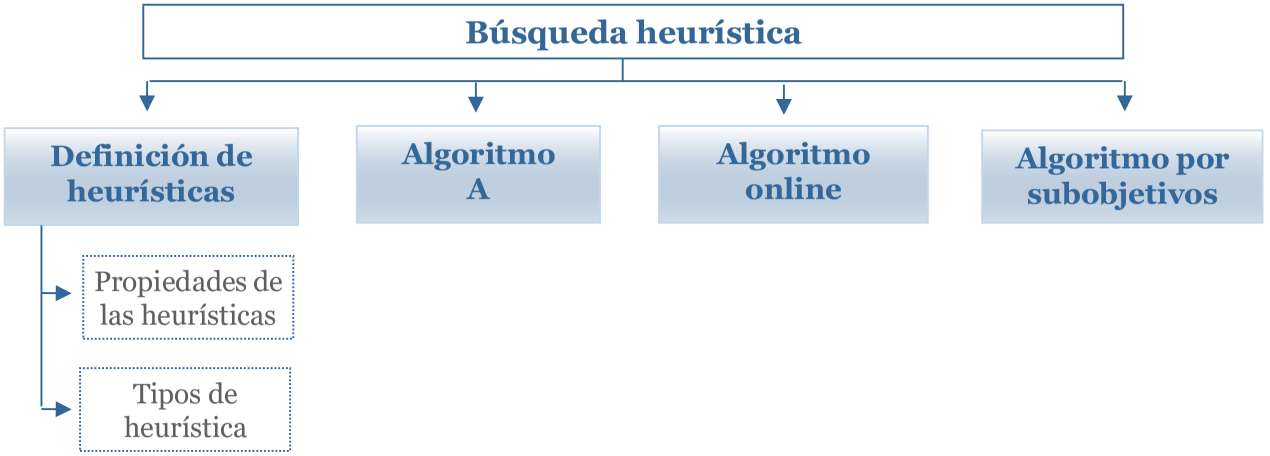
6.6. Referencias bibliográficas 21

Lo + recomendado 22

+ Información 23

Test 24

Esquema



Ideas clave

6.1. ¿Cómo estudiar este tema?

Para estudiar este tema lee las **Ideas clave** que encontrarás a continuación.

En este tema veremos cómo mejorar las búsquedas en entornos más complejos, en los que la expansión de nodos, debido a la complejidad del entorno, puede suponer problemas para realizar la búsqueda.

Exploraremos inicialmente el concepto de **heurística**, como una formalización de la experiencia que se le puede proporcionar a un agente que debe resolver un problema y que le permite guiarse a través del espacio de estados.

Con los conceptos de heurística definiremos tres nuevos modelos de búsqueda en espacios no estructurados, que permitirán acelerar el proceso de exploración incluso en entornos donde la restricción temporal puede suponer un problema muy restrictivo.

En este tema, nos volveremos a apoyar en el libro de Russell y Norvig: *Inteligencia artificial: un enfoque moderno* (2004).

Así, veremos los algoritmos de:

- Búsqueda A*.
- Búsqueda con horizonte limitado.
- Búsqueda online.
- Búsqueda por subobjetivos.

6.2. Tipos de heurísticas

El término «heurística» (del griego: *heuriskein*) significa «encontrar», «descubrir». La heurística surge resolviendo problemas y observando cómo se han resuelto otros.

Podemos definir la heurística como la manera de alcanzar la solución de problemas a través de la evaluación de los progresos alcanzados durante la búsqueda del resultado definitivo.

La heurística es una capacidad típica del ser humano, que le lleva a innovar para alcanzar unos objetivos. En inteligencia artificial, muchos de los algoritmos que se emplean son heurísticos o utilizan reglas heurísticas.

El método heurístico, ya que, como comentamos, se basa en resultados anteriores para lograr nuevos objetivos, puede devolver soluciones verdaderas o falsas, aunque esté correctamente aplicado. En inteligencia artificial, y en las ciencias de la computación en general, el método heurístico se usa en determinadas circunstancias, únicamente cuando no hay una solución óptima con una serie de restricciones de partida.

Los programas heurísticos actúan encontrando algoritmos que proporcionen tiempo de ejecución y soluciones adecuadas. Por ejemplo, en juegos que intentan predecir lo que va a hacer el usuario. Para ello, se basan en la experiencia y en lo que ha hecho el usuario en otras ocasiones.

Un ejemplo de algoritmo heurístico empleado en inteligencia artificial es el encargado de determinar si un email debe ser catalogado como *spam* o como no *spam.* Y es que las reglas, si se emplean de manera independiente, pueden provocar errores de clasificación, pero al ser utilizadas en conjunto, cuando se utilizan muchas reglas heurísticas a la vez, la solución mejora, siendo más robusta y aceptable.

En definitiva, la **heurística en inteligencia artificial representa el conocimiento extraído de la experiencia dentro del dominio del problema**.

Diferenciamos dos tipos de heurística:

- Interpretación «fuerte»: pensada para facilitar la resolución del problema, pero no nos garantiza la resolución del mismo, ni su completitud ni su optimalidad. Es como una «regla de tres» para solucionar un problema.
- Interpretación «débil»: buscamos aplicar un método riguroso junto a la información heurística para guiar el proceso de búsqueda. Queremos mejorar el **rendimiento medio** de un método de resolución de problemas, pero no garantiza una mejora en el peor caso.

El concepto de información heurística que aplicaremos a un método de exploración se basa en las **funciones heurísticas**. Estas funciones para búsqueda en el espacio de estados estiman de adecuación de un nodo para ser expandido y son, por tanto, métodos de búsqueda basados en «el mejor primero», eligiendo el nodo más prometedor para expandir. Una definición simplista es interpretar una heurística como la «distancia» que separa a un nodo de la meta de modo aproximado.

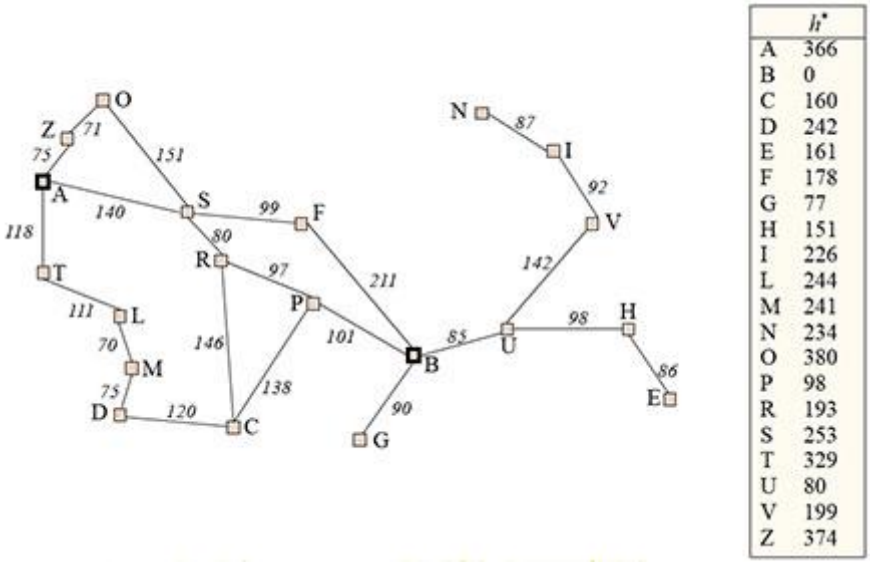
$h : N \longrightarrow \aleph$	Coste *real* desde el nodo *n* hasta la meta más cercana.
$h^* : N \longrightarrow \aleph$	Función heurística que estima el valor de *h(n).*

Tabla 1. Funciones de coste real y de coste estimado (heurística)

Función heurística optimista

Definiremos una función heurística h^* como optimista si $h^*(n) \leq h(n)$ para todo n. Por ejemplo, consideramos como función heurística para el mundo de los bloques el número de bloques **descolocados**. O, en el caso de enfrentarnos a un problema de

encontrar rutas en una red de carreteras, por ejemplo, estimaríamos la distancia en **línea recta** hasta un nodo meta (ver siguiente imagen).



	h^*
A	366
B	0
C	160
D	242
E	161
F	178
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	98
R	193
S	253
T	329
U	80
V	199
Z	374

Figura 1. Ejemplo esquemático de un grafo que representaría una red de carreteras y su coste estimado para ir hasta el nodo B.

Cuestiones

- ¿Cómo se pueden encontrar funciones heurísticas **optimistas y/o consistentes**?
- ¿Cómo se puede distinguir entre «buenas» y «malas» funciones heurísticas?

Diseño de funciones heurísticas

Para el diseño de estas funciones, nos plantearemos el concepto de **problemas relajados**, es decir, aquellos en los que tenemos el mismo conjunto de estados (estado inicial, estado meta, etc.), pero que presentan **menos precondiciones** para cada operador. En ellos, por tanto, todas las soluciones al problema original también lo son al problema relajado y posiblemente a algunos más.

Partiremos de la idea de usar el coste exacto $h(n)$ de llegar desde el estado n a un nodo meta en el problema relajado como valor heurístico $h^*(n)$ en el problema original. Por construcción, una función heurística h^* así construida es optimista.

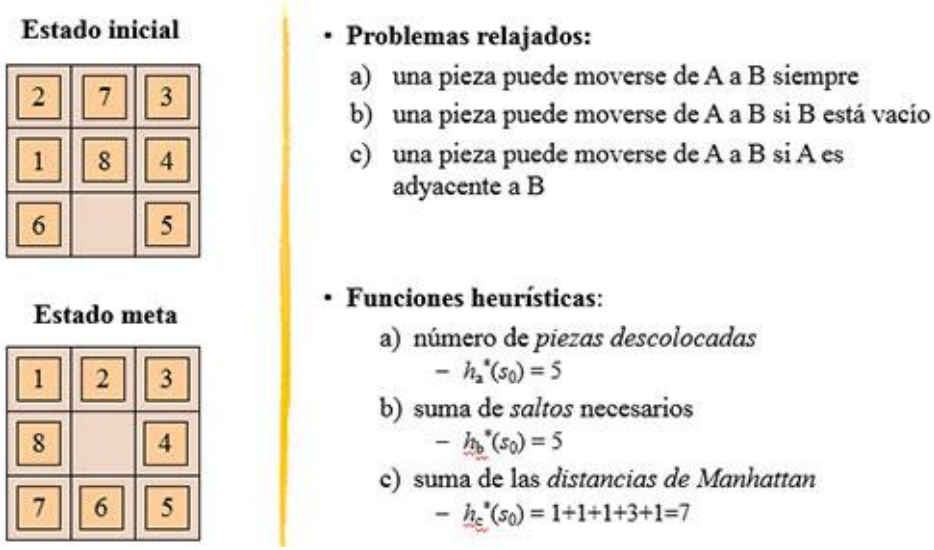


Figura 2. Ejemplo de problemas relajados.

Funciones heurísticas y su calidad

Si tenemos dos funciones heurísticas, $h1^*$ y $h2^*$ optimistas, diremos que $h1^*$ es más informada que $h2^*$ si para todo nodo n se cumple que $h1^*(n) \geq h2^*(n)$.

De este modo, si $h1^*$ es más informada que $h2^*$, entonces el algoritmo que emplee $h2^*$ expande al menos tantos nodos como aquel que emplee $h1^*$.

Por tanto, será preferible elegir aquellas funciones que presenten valores de h^* grandes, siempre que se mantenga optimista. Si hay varias funciones heurísticas optimistas, elegiríamos aquellas en cada caso que tengan mayor valor en cada momento.

$$h^*(\mathrm{n}) = \max h_1^*(n), h_2^*(n), \ldots, h_m^*(n))$$

6.3. Búsqueda A*

La idea es orientar la búsqueda de una solución en la que las acciones no tienen el mismo coste y desconocemos, *a priori*, el coste de llegar hasta un estado meta, pero podemos estimar, por medio de una función heurística, el coste restante que nos queda desde el nodo actual para alcanzar el nodo meta. Por tanto, nuestro objetivo es minimizar el coste estimado de un camino en el árbol de búsqueda, combinando el coste de llegar al nodo *n* (conocido exactamente por **g(n)**) y el coste aproximado de llegar de este nodo hasta la meta por medio de la función heurística h^*.

Para nuestro caso, estableceremos una función heurística que subestime el coste real $f(n)$ que nos resta para llegar a la meta desde un nodo cualquiera n:

$-f(n) = g(n)+h(n)$: coste mínimo que pasa por n, es decir, el **coste real** del plan.

Para ello, emplearemos la función heurística f^*, que estimará una aproximación de $f(n)$:

$$f^*(n) = g(n)+h^*(n)$$

Durante el proceso de búsqueda, emplearemos la estrategia de elegir, entre las hojas del árbol de búsqueda, aquella con un valor f^* mínimo.

El **algoritmo A*** se basa en la búsqueda general, pero deberemos añadir el valor g a cada nodo expandido y ordenar la lista abierta en base a los valores crecientes de f^*. Por tanto, añadiremos los nuevos nodos en la lista abierta según sus valores crecientes de f^*.

```
{A*}
abierta ← s0
Repetir
    Si vacío?(abierta) entonces
        devolver(negativo)
    nodo ← primero(abierta)
    Si meta?(nodo) entonces
        devolver(nodo)
    sucesores ← expandir(nodo)
    Para cada n∈sucesores hacer
        n.padre ← nodo
        ordInsertar(n,abierta, f*)
    Fin {repetir}
```

Figura 3. Algoritmo de búsqueda A*.

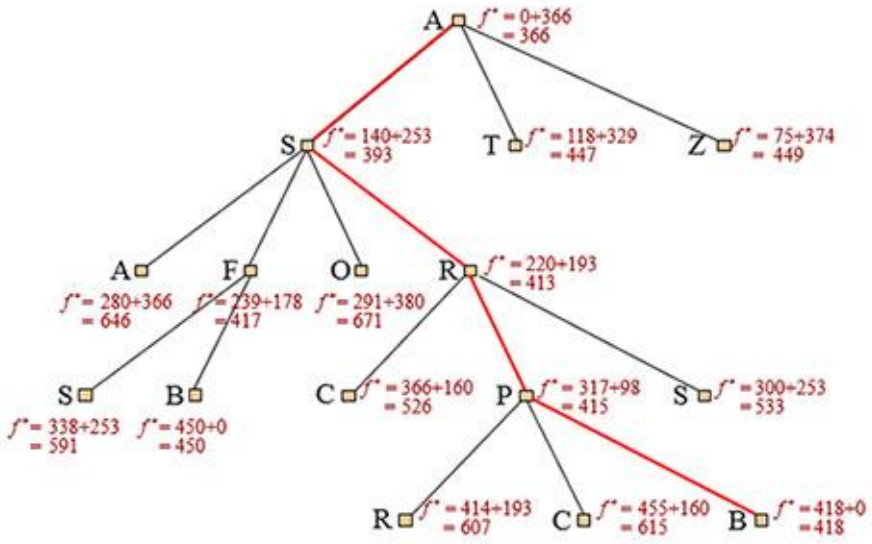


Figura 4. Ejemplo de búsqueda A*.

La **estrategia A*** continúa mientras se produce un crecimiento en el valor de f^* a lo largo de los caminos del árbol de búsqueda. Por tanto, si nos alejamos de la meta, g crece y h^* crece, por lo que f^* crece en gran medida; en cambio, si nos acercamos a la meta, el valor de f^* crece poco porque g crece mientras que h^* disminuye.

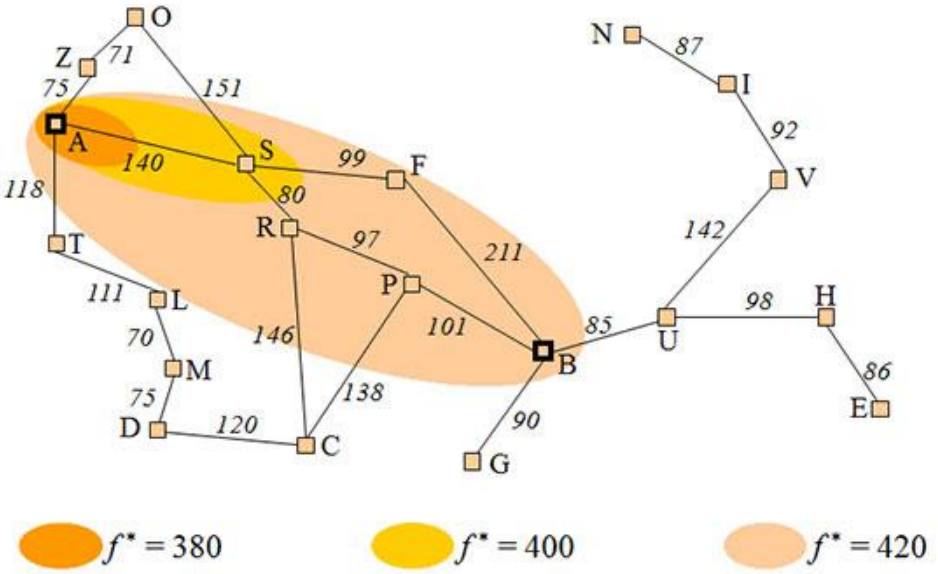


Figura 5. Ejemplo de evolución de los valores de f* a lo largo de una búsqueda.

Si lo comparamos con la búsqueda de coste uniforme, los valores de g crecen a lo largo de todos los caminos del árbol de búsqueda porque en cada paso se suma el coste de un operador (que es un número natural positivo).

En el ejemplo anterior, los valores de f^* también crecen a lo largo de todos los caminos del árbol de búsqueda. Pero esto no siempre ha de ser así:

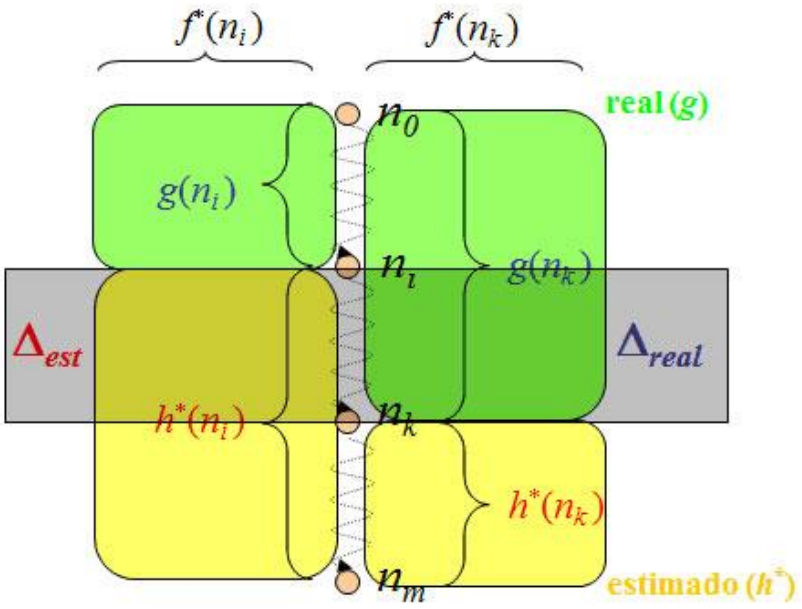


Figura 6. Comportamiento esperado del aumento del coste real respecto al coste estimado.

En un camino del árbol de búsqueda, nk es sucesor (no necesariamente directo) de ni, entonces el valor de $f^*(n_i)$ contiene una **estimación** del coste de llegar de n_i a n_k (D_{est}) y $f^*(n_k)$ el coste real (D_{real}). Es decir, se puede obtener el valor de $f^*(n_k)$ a partir de $f^*(n_i)$, sumando D_{real} y restando D_{est}. Es posible que $f^*(n_k) < f^*(n_i)$, *si* $D_{est} > D_{real}$.

La función heurística en la que nos apoyemos debe ser optimista.

Cota de f* con función heurística optimista

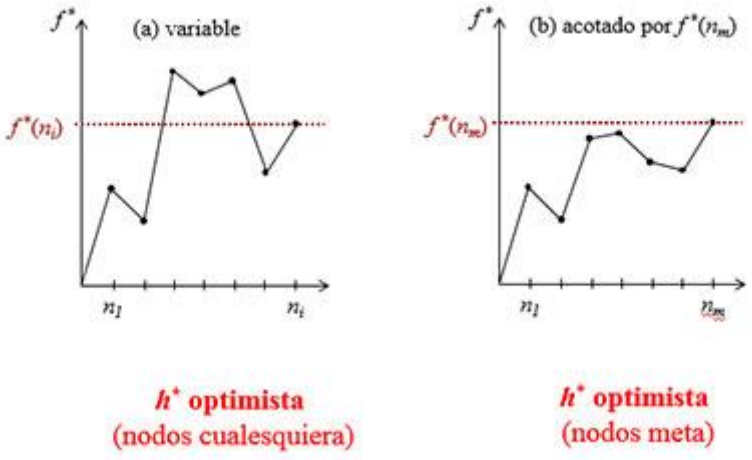


Figura 7. Comportamiento en una función heurística para nodos, meta o intermedios.

Análisis del Algoritmo A*

- Es óptimo: si h^* es optimista.
- Es completo.

Presenta una complejidad dependiente del número de nodos expandidos, que depende, a su vez, de la precisión de h^*, de tal modo que si tuviésemos que la función heurística fuese de perfecta información $(h^*(n) = h(n)$ *para todos los nodos* $n)$, la complejidad sería lineal (sin contar la complejidad de computar h^*). Mientras que, si

carecemos de información, *(h*(n) = 0 para todos los nodos n),* la búsqueda A* degenera en la búsqueda de coste uniforme.

Aun así, suele haber una mejora notable en comparación con métodos no informados.

Por ejemplo, el número de nodos expandidos en el problema del 8-puzzle, suponiendo que la profundidad de la solución se encontrase tras d niveles.

d	B. no informada (prof. iterativa)	$A^*(h_a)$	$A^*(h_c)$
2	10	6	6
4	112	13	12
6	680	20	18
8	6.384	39	25
10	47.127	93	39
12	3.644.035	227	73
14	—	539	113
16	—	1.301	211
18	—	3.056	363
20	—	7.276	676
22	—	18.094	1.219
24	—	39.135	1.641

Figura 8. Media estimada para cien ejecuciones. Fuente: basado en Russell y Norvig (2004)

6.4. Búsqueda por subobjetivos

El uso de heurísticas puede reducir drásticamente la complejidad de los algoritmos de búsqueda. Por ello es una estrategia que debe considerarse en muchos problemas, ya sea con una aplicación débil o con una aplicación más fuerte.

El Algoritmo A* emplea una aplicación «débil» de información heurística. Se usan heurísticas para guiar la búsqueda, no para reducir el espacio de posibles soluciones. Se mantienen propiedades de completitud y *optimalidad*.

En muchos casos, se requiere/prefiere una aplicación «fuerte» de heurísticas que reduce el espacio de búsqueda de forma efectiva (en anchura o profundidad), aunque se corra el riesgo de no explorar posibles soluciones, dado que estas son poco probables. Así, es posible que no se encuentre la mejor solución (no ser óptimos). Pero el objetivo es encontrar soluciones «buenas» (no necesariamente óptimas) y mejorar el rendimiento de forma substancial/*optimalidad* y completitud no garantizados.

La **búsqueda por subobjetivos** tiene como objetivo principal **reducir la complejidad de un problema de búsqueda**. La idea se fundamenta en disminuir la profundidad de las búsquedas, subdiviendo el problema en varios problemas más pequeños mediante una heurística fuerte. Intenta determinar una secuencia de estados intermedios ($i_1, i_2,..., i_n$) que con mayor probabilidad van a encontrarse en el camino óptimo. Con estos estados intermedios, realizará búsquedas con un método base (amplitud, prof., prof. iterativa...) desde el estado inicial s_0 *a* i_1, *de* i_1 *a* i_2, etc. y de i_n al estado meta s_m.

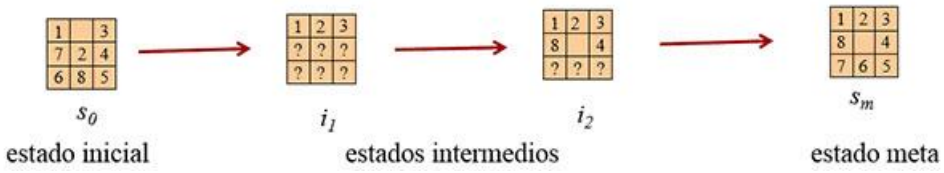


Figura 9. Subobjetivos para el problema de Puzzle-8.

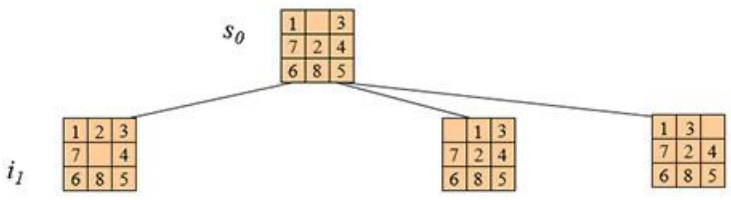


Figura 10. Método base: búsqueda en amplitud camino de s_0 a i_1.

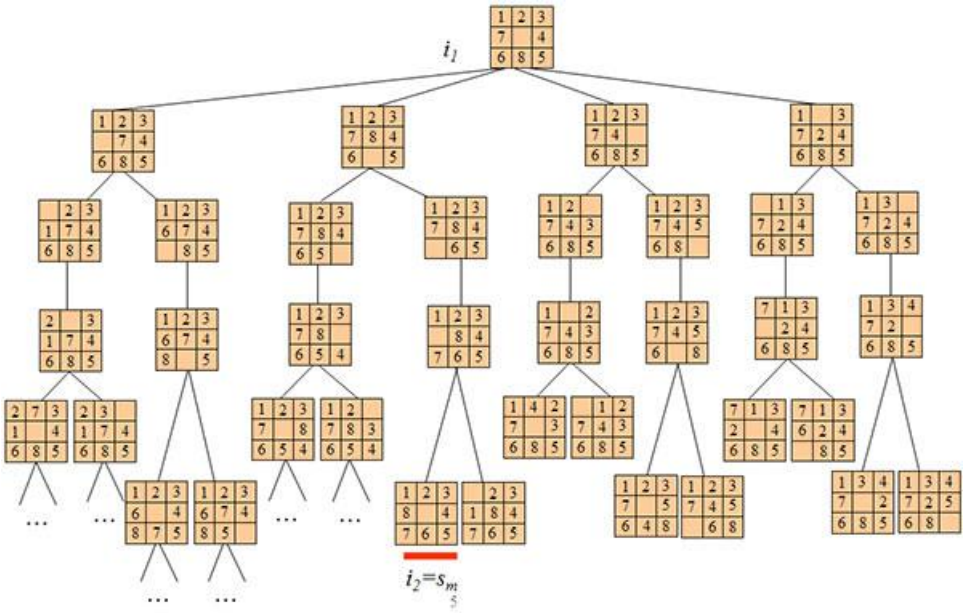


Figura 11. Camino de i_1 a i_2.

Complejidad de búsqueda por subobjetivos

En esta técnica nos encontramos con que la complejidad en tiempo y espacio depende de condiciones diferentes:

- Complejidad en tiempo: nodos expandidos.
- Complejidad en espacio: nodos abiertos.

En general, si la elección de subobjetivos es buena, se obtiene una mejora en la complejidad, siempre que los caminos entre los pares de subojetivos estén «más cortos» que la solución global.

Por ejemplo, para un problema cuyo factor de expansión fuese $b=3$, la profundidad de la solución $d=20$ y creásemos $x=4$ subobjetivos de profundidad $d'=5$ (el camino encontrado tiene una longitud de 25). Y empleando como técnica básica un algoritmo de búsqueda en amplitud.

		General		Ejemplo	
		Bús. prof.	Bús. subob.	Bús. prof.	Bús. subob.
mejor caso:	- tiempo:	$\frac{b^d - 1}{b - 1}$	$(x+1)\frac{b^{d'} - 1}{b - 1}$	1.743.392.200	605
	- espacio:	$\frac{b^{d+1} - 1}{b - 1}$	$(x+1)\frac{b^{d'+1} - 1}{b - 1}$	5.230.176.601	1820
peor caso:	- tiempo:	$\frac{b^{d+1} - 1}{b - 1} - 1$	$(x+1)\frac{b^{d'+1} - 1}{b - 1} - 1$	5.230.176.600	1815
	- espacio:	$\frac{b^{d+2} - 1}{b - 1} - b$	$(x+1)\frac{b^{d'+2} - 1}{b - 1} - b$	15.690.529.801	5450

Figura 12. Estimaciones de complejidad algorítmica.

Análisis de búsqueda por subobjetivos:

Permite reducir de forma substancial la complejidad, especialmente en problemas de planes muy largos. La búsqueda por subobjetivos no es necesariamente completa ni óptima.

Por tanto, es completo si el método base es completo y los subobjetivos se encuentran en el camino que nos permite llegar desde el estado inicial al final. Asimismo, será optimo si el método base es óptimo y los subobjetivos se encuentran ordenados de tal modo que su camino sea el mínimo posible dentro del camino solución.

En el caso de utilizar búsquedas heurísticas como método base (por ejemplo, A*), se puede alcanzar una reducción aún mayor de la complejidad, pero puede llegar a ser necesario especificar distintas funciones heurísticas para cada subobjetivo.

6.5. Búsqueda online

En algunos entornos no es posible/útil encontrar un plan completo. En entornos no deterministas/dinámicos, no se pueden determinar los resultados de las acciones, o el entorno o el estado meta cambia. Por ejemplo: seguir un objeto en movimiento (escalera de Hogwarts).

Tampoco es posible encontrar un plan completo en entornos (parcialmente) inaccesibles, donde se desconoce el entorno (estados existentes, resultados de las acciones). Por ejemplo: laberinto.

Surgen de la necesidad que tienen los agentes reactivos de encontrar una solución en un tiempo muy corto. Esta falta de tiempo para encontrar la solución hace que en muchos casos baste con encontrar una buena acción, no un plan de acción completo bueno.

La idea básica es combinar un paso de búsqueda con un paso de actuación.

Búsqueda online	Búsqueda offline
Repetir: 1. Percibir entorno. 2. Elegir la «mejor» acción. 3. Realización acción. Fin repetir.	1. Percibir entorno. 2. Buscar plan. 3. Realizar plan.

Tabla 2. Comparativa entre la búsqueda online y offline.

En esta sección, vamos a ver:

- Búsqueda por ascenso de colinas.
- Búsqueda por horizonte.
- Optimización mediante búsqueda online.

Búsqueda por ascenso de colinas *(hill climbing)*

Tiene como idea inicial disminuir la profundidad del árbol de búsqueda e intentar llevar a cabo la acción que parece más prometedora en cada momento, repitiendo el ciclo percepción/acción de forma continua.

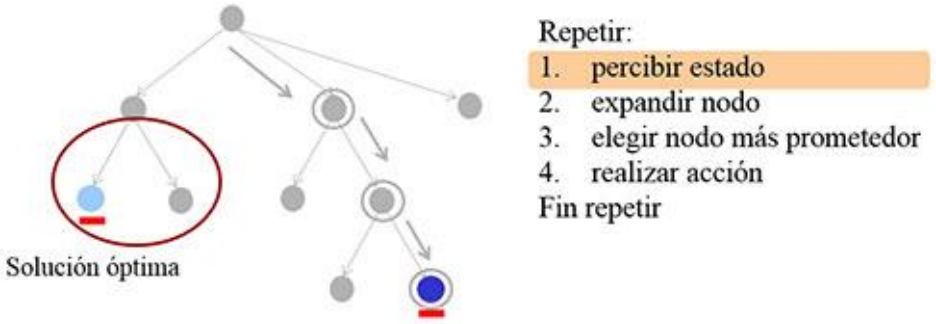


Figura 13. Comportamiento de la búsqueda por ascenso de colinas.

El algoritmo de búsqueda por ascenso de colinas emplea una función heurística para calcular la distancia estimada al nodo meta más cercano desde el nodo n, $h^*(n)$; por otro lado, debe devolver el estado actual de problema por medio de un mecanismo de percepción *(percibir(entorno)).* En un estado determinado por el nodo n, debe evaluar cuál de las posibles acciones es la mejor por medio de la estimación heurística de las opciones presentes *(evaluar(n,h*)).* Una vez decidida cuál es la mejor acción, la ejecutaremos dentro del entorno.

{búsqueda ascenso de colinas}
Repetir
nodo ← *percibir(entorno)*
Si *meta?*(nodo) **entonces**
devolver(positivo)
sucesores ← *expandir*(nodo)
Si *vacia?*(sucesores) **entonces**
devolver(negativo)
mejor ← $\arg\min_{n \in sucesores}[evaluar(n,h^*)]$
a ←*acción(n,mejor)*
ejecutar(a, entorno)
Fin {repetir}

Figura 14. Algoritmo de la búsqueda por ascenso de colinas.

Análisis de búsqueda por ascenso de colinas

En general, **no se puede asegurar *optimalidad* ni completitud**. Pero suele producir resultados mejores, sobre todo en los casos en los que no existan bucles y la función

heurística sea buena. Si h^*es completa $(h^*=h)$, entonces es **óptimo y completo**. Hay que vigilar la eliminación de ciclos y eliminar los estados repetidos.

Búsqueda con horizonte

Hacemos una expansión de los nodos hasta la profundidad k(horizonte) y realizamos aquella acción que es más prometedora de todas las acciones de ese nivel, repitiendo de forma continua hasta que encontramos una solución dentro del horizonte de búsqueda.

El algoritmo general arranca en un estado s, desde el cual aplicamos una búsqueda no informada de profundidad k, que puede ser suspendida en los nodos meta. De las hojas (H) adquiridas en el nivel k, obtenemos la mejor empleando una función heurística h^*, de tal modo que optaremos por aquella acción que nos lleve a la hoja de menor coste estimado. Repetiremos este proceso hasta alcanzar la meta.

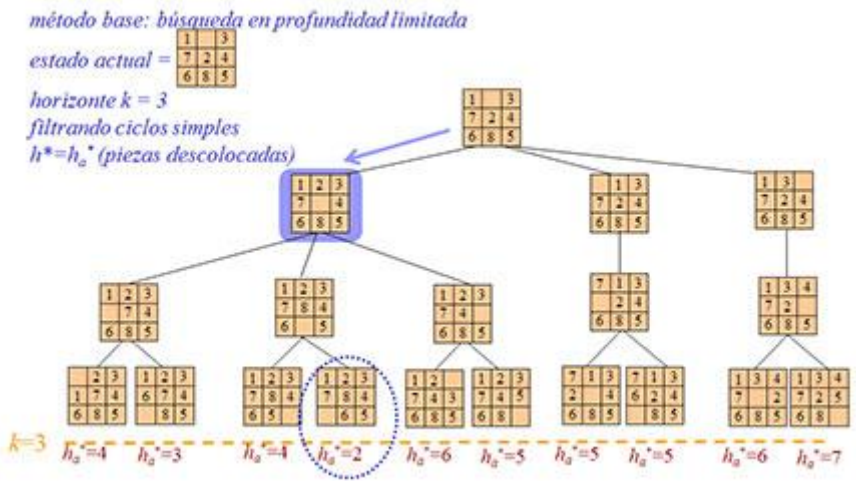


Figura 15. Ejemplo de búsqueda en profundidad limitada (paso 1).

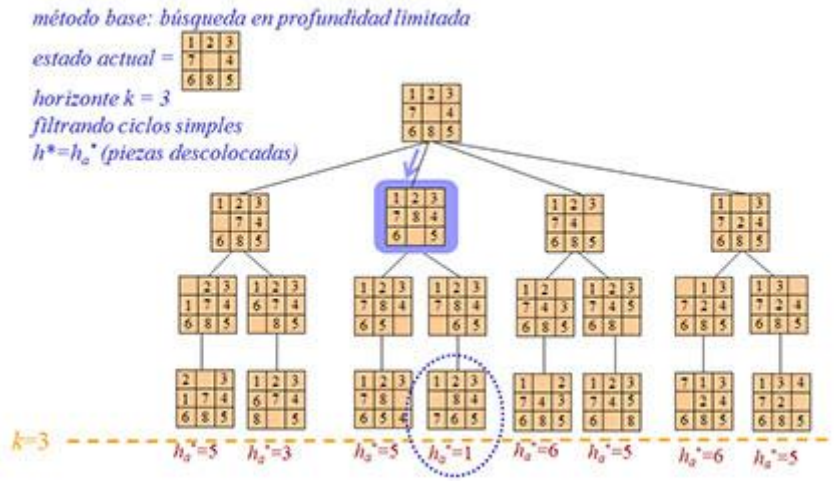


Figura 16. Ejemplo de búsqueda en profundidad limitada (paso 2).

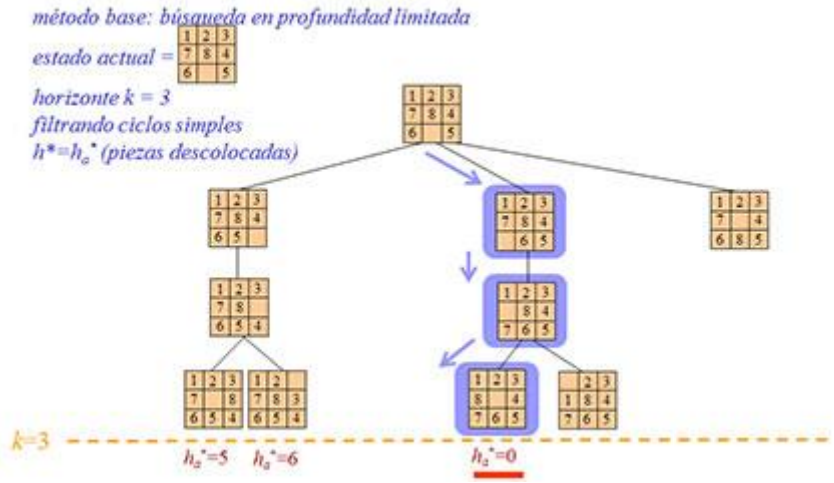


Figura 17. Ejemplo de búsqueda en profundidad limitada (paso 3, final).

Análisis de búsqueda con horizonte

En general, no se puede asegurar *optimalidad* ni completitud, dado que esta está ligada directamente con la función heurística. En el caso de tener una h^* **completa** $(h^*=h)$**, entonces es óptimo y completo**.

La búsqueda por ascenso de colinas es un subtipo de la búsqueda con horizonte (horizonte = 1). El aumento del horizonte de exploración permite evitar bucles de longitud menor que k.

Optimización mediante búsqueda online

Un subtipo de problemas a los que se puede aplicar con éxito las búsquedas online (horizonte/ascenso de colinas) es el de los problemas de optimización, es decir, aquellos en los que tenemos una función objetivo que deseamos alcanzar. En este tipo de problemas no necesitamos obtener el valor máximo/mínimo de la función que deseamos alcanzar, sino que nos basta con el valor que más se aproxime a dicho óptimo.

El camino para llegar al estado buscado puede ser irrelevante (coste 0). Algunos ejemplos: juegos lógicos y de configuración, n-Reinas.

En los problemas de optimización con funciones heurísticas estimas el coste del camino que parte del estado actual y va hasta el nodo meta más próximo, midiendo la calidad de un nodo con respecto a una función de evaluación objetivo.

6.6. Referencias bibliográficas

Russell, S. y Norvig, P. (2004). *Inteligencia Artificial: Un Enfoque Moderno*. Madrid: Pearson Educación.

Lo + recomendado

No dejes de leer

Enfoque de búsqueda en tiempo real

Fernández, M. O. (1998). *Algoritmos de búsqueda heurística en tiempo real. Aplicación a la navegación en los juegos de vídeo.* Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina.

En este trabajo encontrarás información sobre cómo plantear una búsqueda online en tiempo real en un entorno como podría ser el de los videojuegos.

Accede al artículo a través del aula virtual o desde la siguiente dirección web:
http://www.exa.unicen.edu.ar/catedras/aydalgo2/docs/TFca06aCompleto.pdf

+ Información

A fondo

Sesgos cognitivos

González, J. (1 de enero de 2013). Heurísticos y sesgos cognitivos: los atajos de la mente.

Aquí se presenta una discusión sobre la integración de funciones heurísticas y de sesgo cognitivo en el que nos basamos a la hora de realizar estimaciones optimistas.

Estos sesgos son importantes a la hora de evaluar nuestra neutralidad en las estimaciones, tanto para realizar algoritmos de búsqueda como para modelizar cualquier tipo de toma de decisiones.

Accede al artículo a través del aula virtual o desde la siguiente dirección web: http://jesusgonzalezfonseca.blogspot.com.es/2013/01/heuristicos-y-sesgos-cognitivos-los.html

Test

1. En el algoritmo A* se usan heurísticas para:

A. Reducir el espacio de posibles soluciones.

B. Guiar la búsqueda.

C. Se emplea una aplicación débil de información heurística.

2. La aplicación fuerte de heurísticas hace que:

A. Se reduzca el espacio de búsqueda.

B. Siempre encuentre la mejor solución.

C. Tiene como objetivo encontrar soluciones buenas, aunque no sean necesariamente óptimas.

3. El objetivo de la búsqueda por subobjetivos es:

A. Ampliar la complejidad de un problema de búsqueda.

B. Reducir la complejidad de un problema de búsqueda.

C. Subdividir el problema en varios problemas pequeños mediante la aplicación de una heurística fuerte.

4. En cuáles de los siguientes entornos no es posible o útil encontrar un plan completo:

A. Entornos no deterministas.

B. Entonos inaccesibles.

C. Agentes reactivos que actúan de forma rápida o con objetivo continuo.

5. La idea básica de la búsqueda online es:

A. La combinación de un paso de búsqueda con un paso de actuación.

B. Realizar solo pasos de búsqueda de manera sucesiva.

C. Ninguna de las respuestas anteriores es correcta.

6. La idea que subyace en la búsqueda por ascenso de colinas es:

A. Reducir profundidad del árbol expandido y realizar siempre la acción más prometedora.

B. Expandir el árbol solo un nivel, realizar la acción hacia el mejor nodo y repetir el ciclo percepción-acción de manera continuada.

C. Ninguna de las respuestas anteriores es correcta.

7. En la búsqueda por ascenso de colinas, se puede conseguir completitud si:

A. El conjunto de posibles estados es infinito.

B. El conjunto de posibles estados es finito.

C. Se eliminan todos los estados por los que se ha ido pasando.

8. En la búsqueda con horizonte:

A. La primera acción de un plan que lleva a un nodo con una evaluación heurística óptima tiene una buena probabilidad de pertenecer al camino óptimo.

B. Son reiteradas acciones de un plan las que llevan a un nodo con una evaluación heurística óptima.

C. Asegura que hay *optimalidad* y completitud.

9. En un problema de optimización:

A. El camino para llegar al estado siempre es relevante.

B. No busca un estado exacto, sino que se acerque al máximo objetivo.

C. El objetivo es encontrar el mejor estado según una función objetivo.

10. La búsqueda online para optimización es útil para problemas con:

A. Número infinito de estados.

B. Número finito de estados.

C. Tanto para número infinito como finito de estados.