

ALGORITMOS DE BUSQUEDA- TECNICAS INFORMADAS

Inteligencia Artificial



BUSQUEDAS INFORMADAS

Las técnicas informadas utilizan pistas específicas del dominio sobre la ubicación de los objetivos

Tienen la capacidad de encontrar soluciones de manera más eficiente que una estrategia no informada o ciega. Las pistas vienen en forma de una función heurística, denotada como $h(n)$.

Por ejemplo, en problemas de búsqueda de rutas, podemos estimar la distancia desde el estado actual a un objetivo calculando la distancia en línea recta en el mapa entre los dos puntos.**(Algoritmo de Maze con Euclidean distance)**.



BUSQUEDAS INFORMADAS- Greedy Best-First

BUSQUEDAS INFORMADAS-Greedy Best First

Es una forma de aplicación de la búsqueda Best-First Search que expande primero el nodo con el valor **$h(n)$** más bajo (el nodo que parece estar más cerca del objetivo), con el argumento de que esto probablemente llevará a una solución rápidamente.

$$f(n) = h(n).$$

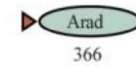
BUSQUEDAS INFORMADAS-Greedy Best First

Usemos la heurística de distancia en línea recta, que llamaremos h_{SLD} .

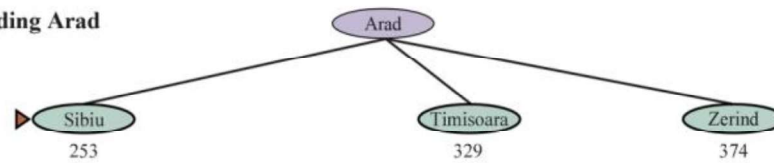
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

BUSQUEDAS INFORMADAS-Greedy Best First

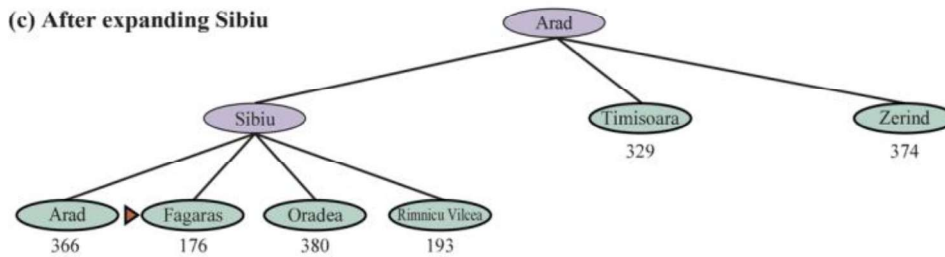
(a) The initial state



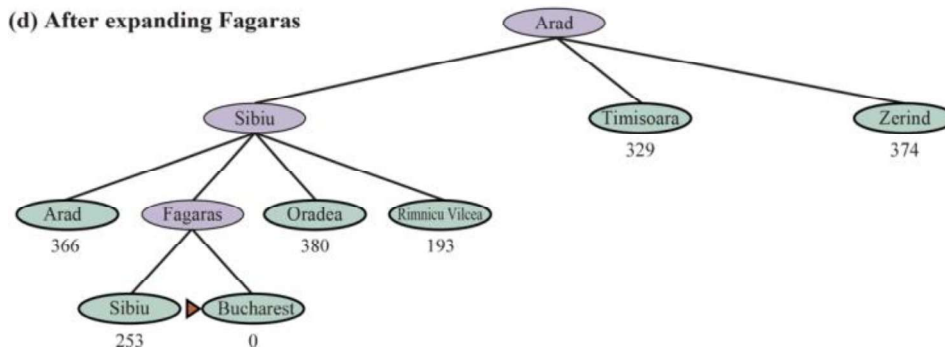
(b) After expanding Arad



(c) After expanding Sibiu



(d) After expanding Fagaras





BUSQUEDAS INFORMADAS- A* Search

BUSQUEDAS INFORMADAS- A* Search

El algoritmo de búsqueda informada más común es la búsqueda "A-star search", una búsqueda best-first que utiliza la función de evaluación:

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

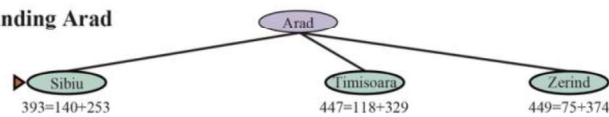
Donde $g(n)$ es el costo del camino desde el estado inicial hasta el nodo n , y $h(n)$ es el costo estimado del camino más corto desde n hasta un estado objetivo

BUSQUEDAS INFORMADAS- A* Search

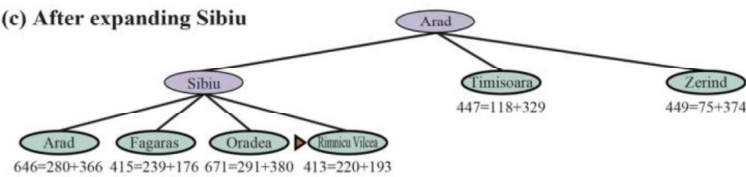
(a) The initial state



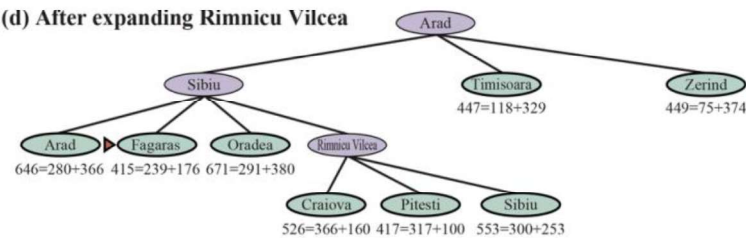
(b) After expanding Arad



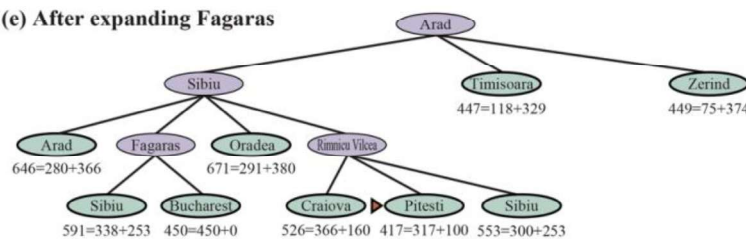
(c) After expanding Sibiu



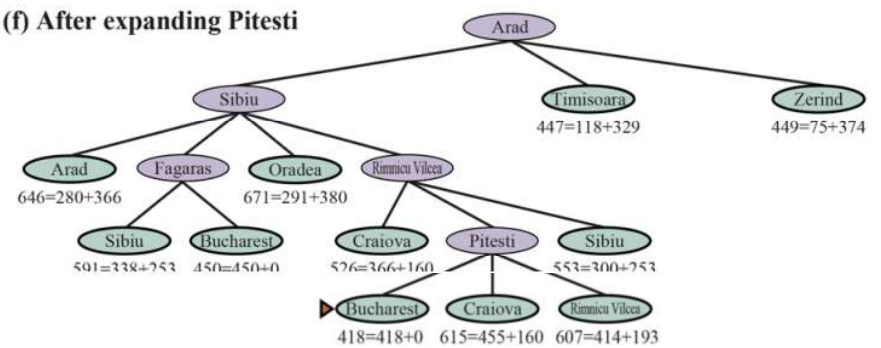
(d) After expanding Rimnicu Vilcea



(e) After expanding Fagaras



(f) After expanding Pitesti



BUSQUEDAS INFORMADAS- A* Search

CARACTERISTICAS:

La búsqueda A* es **completa**.

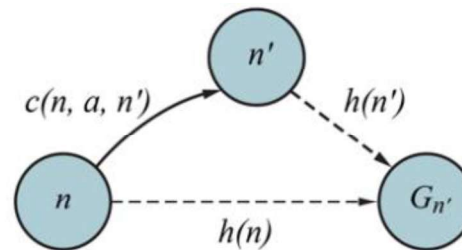
Que sea **optima** en costo depende de:

Admisibilidad: Dado que se cumpla que $h(n) \leq h^*(n)$,

Donde $h^*(n)$ es el costo real mínimo desde n hasta el objetivo.

Consistencia: Una heurística $h(n)$ es consistente si, para cada nodo n y cada sucesor n' de n generado por una acción a, tenemos:

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



BUSQUEDAS INFORMADAS- A* Search

CONCLUSIÓN:

A* Search puede ser completa, óptima en costo y óptimamente eficiente

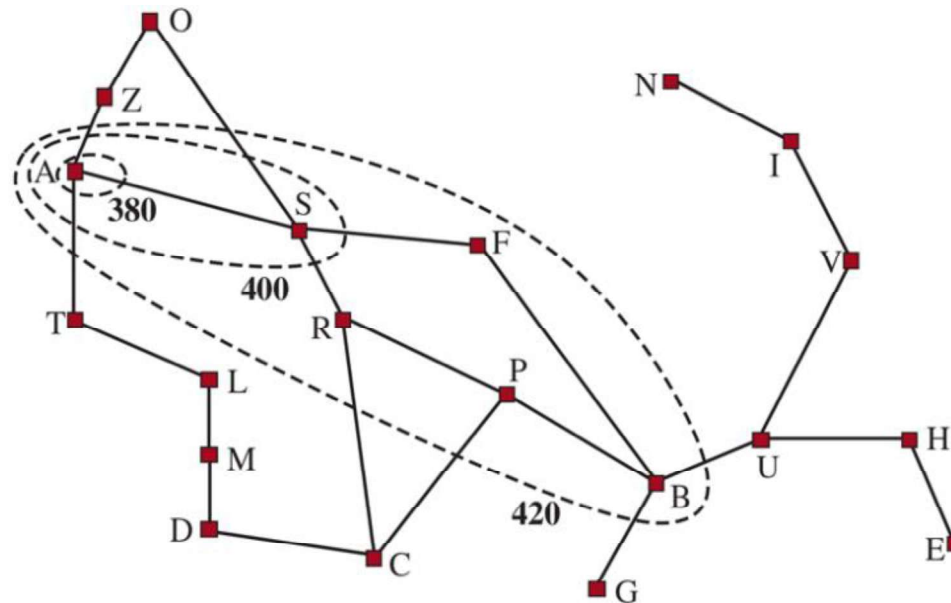
El inconveniente es que, para muchos problemas, el número de nodos expandidos puede ser exponencial en relación con la longitud de la solución.




BUSQUEDAS INFORMADAS- Search Countours

BUSQUEDAS INFORMADAS- Search Countours

Una forma útil de visualizar una búsqueda es dibujar contornos en el espacio de estados, de manera similar a los contornos en un mapa topográfico





BUSQUEDAS INFORMADAS- Inadmissible heuristics and weighted A*

BUSQUEDAS INFORMADAS- Inadmissible heuristics

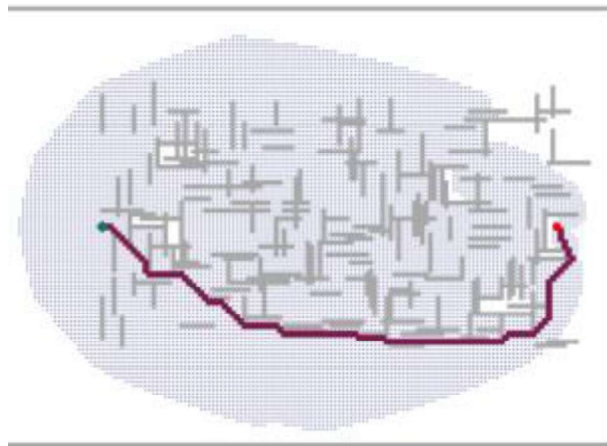
La búsqueda A* tiene muchas cualidades positivas, pero expande muchos nodos. Podemos explorar menos nodos (utilizando menos tiempo y espacio) si estamos dispuestos a aceptar soluciones que no sean óptimas, pero que sean "suficientemente buenas": lo que llamamos **soluciones satisfactorias**.

Si permitimos que la búsqueda A* utilice una heurística inadmisble (una que puede sobreestimar), podemos reducir la cantidad de nodos expandidos

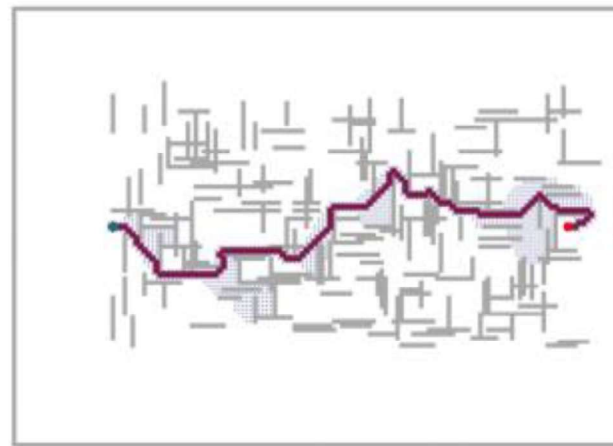
$$f(n) = g(n) + W \cdot h(n) \quad \text{para algún } W > 1$$

BUSQUEDAS INFORMADAS- Inadmissible heuristics

En general, si la solución óptima cuesta C^* , una Weighted A* Search encontrará una solución que cuesta entre C^* and $W \times C^*$; ; pero en la práctica, usualmente obtenemos resultados mucho más cercanos a C^* que a C^* and $W \times C^*$;



(a)



(b)

BUSQUEDAS INFORMADAS- Inadmissible heuristics

A* search:	$g(n) + h(n)$	$(W = 1)$
Uniform-cost search:	$g(n)$	$(W = 0)$
Greedy best-first search:	$h(n)$	$(W = \infty)$
Weighted A* search:	$g(n) + W \times h(n)$	$(1 < W < \infty)$



BUSQUEDAS INFORMADAS- Recursive best-first search

BUSQUEDAS INFORMADAS- Inadmissible heuristics

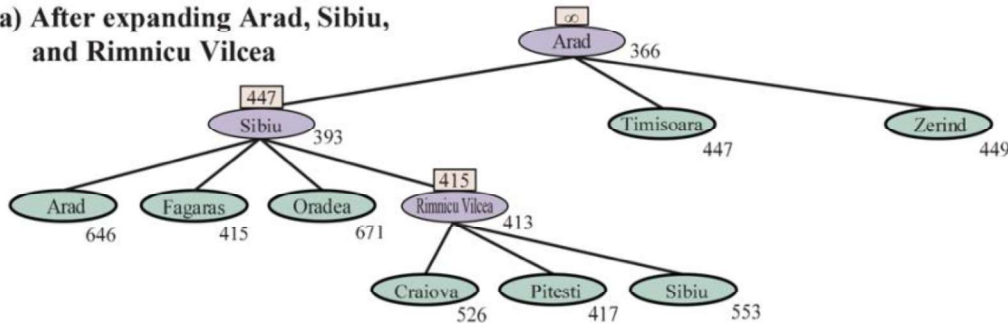
La memoria se divide entre la frontera y los estados alcanzados. En nuestra implementación de búsqueda mejor primero, un estado que está en la frontera se almacena en dos lugares: como un nodo en la frontera (para que podamos decidir qué expandir a continuación) y como una entrada en la tabla de estados alcanzados (para saber si hemos visitado el estado antes).

Recursive Best-First Search (RBFS) es uno de los algoritmos diseñados para conservar el uso de la memoria

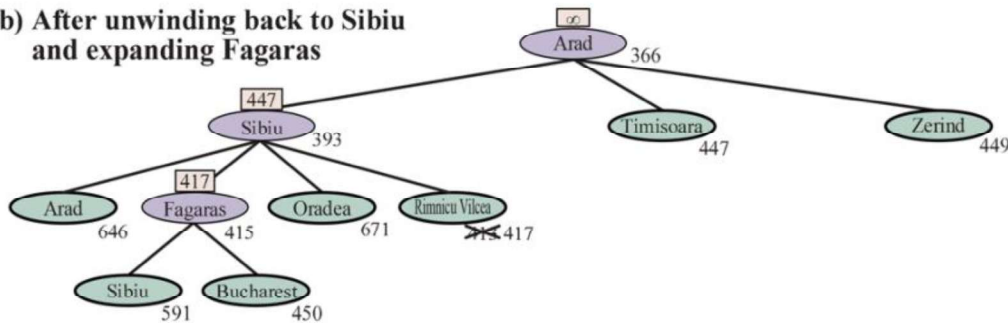
BUSQUEDAS INFORMADAS- Inadmissible heuristics

RBFS se asemeja a una búsqueda recursiva en profundidad, pero en lugar de continuar indefinidamente por el camino actual, utiliza la variable *limit* para hacer un seguimiento del valor f de la mejor ruta alternativa disponible desde cualquier antecesor del nodo actual

(a) After expanding Arad, Sibiu, and Rimnicu Vilcea

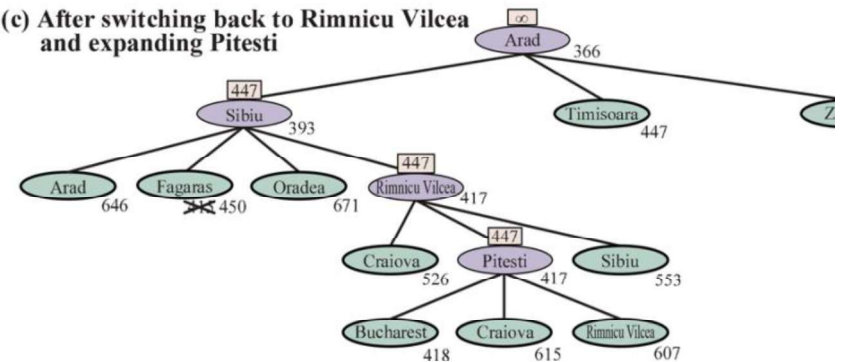


(b) After unwinding back to Sibiu and expanding Fagaras



Vigilada Mineducación

(c) After switching back to Rimnicu Vilcea and expanding Pitesti



BUSQUEDAS INFORMADAS- Inadmissible heuristics

function RECURSIVE-BEST-FIRST-SEARCH(*problem*) **returns** a solution or *failure*
 solution, fvalue \leftarrow RBFS(*problem*, NODE(*problem*.INITIAL), ∞)
return *solution*

function RBFS(*problem, node, f_limit*) **returns** a solution or *failure*, and a new *f*-cost limit
 if *problem*.IS-GOAL(*node*.STATE) **then return** *node*
 successors \leftarrow LIST(EXPAND(*node*))
 if *successors* is empty **then return** *failure, ∞*
 for each *s* **in** *successors* **do** // update *f* with value from previous search
 s.f \leftarrow max(*s*.PATH-COST + *h*(*s*), *node.f*)
 while true do
 best \leftarrow the node in *successors* with lowest *f*-value
 if *best.f* > *f_limit* **then return** *failure, best.f*
 alternative \leftarrow the second-lowest *f*-value among *successors*
 result, best.f \leftarrow RBFS(*problem, best, min(f_limit, alternative)*)
 if *result* \neq *failure* **then return** *result, best.f*