SI516 -A – Sistemas Inteligentes e Innovación

Docente: Ing. Carlos Wilfredo Egüez Terrazas

Métodos algorítmicos para resolución de problemas mediante búsqueda

PRESENTADO POR:

Antonio Miguel Natusch Zarco

Santa Cruz de la Sierra, Bolivia
07 de julio del 2025

1. Problema de búsqueda

1.1. Definición formal

Según Rusell & Norvig (2021, pg. 83), se puede definir formalmente a un problema de búsqueda de la siguiente manera:

- Un conjunto de estados posibles en los que el entorno puede estar denominado **espacio de estado.**
- El **estado inicial** en el que el agente empieza.
- Un conjunto de uno o más **estados meta** donde, a veces, puede haber solo un estado meta, un conjunto pequeño de estados meta alternativos, o una propiedad que es aplicable a muchos estados.
- Las **acciones** disponibles para el agente. Data un estado *s*, ACCIONES(*s*) retorno un conjunto finito de acciones que pueden ser ejecutadas en *s*.
- Un **modelo de transición**, el cual describe qué hace cada acción. RESULTADO(*s, a*) devuelve el estado resultante de efectuar la acción *a* en el estado s.
    - Siguiendo el ejemplo del problema del mapa rumano, RESULTADO(Arad, AZerind) = Zerind; es decir, someter al **estado** de estar en Arad a la **acción** *AZerind* da como resultado llegar al estado Zerind.
- Una **función de costo de acción,** denotada por COSTO-DE-ACCIÓN(*s, a, s'*) cuando estamos programando o *c(s, a, s')* cuando estamos haciéndolo matemáticamente, la cual da el costo numérico de aplicar una acción *a* al estado *s* para llegar al estado *s'*.

Una secuencia de acciones forma un **camino,** y una **solución** es un camino desde el **estado inicial** al **estado meta.** Asumiendo que los costos de acción son acumulados, es decir, el costo total de un camino es la suma de todos los costos de acción individuales, una **solución óptima** tiene el camino con menor costo entre todas las soluciones posibles.

2. Estrategias de búsqueda informada

2.1. Definición

Se le da el nombre de **estrategia de búsqueda informada** a aquella estrategia que hace uso de pistas **específicas del dominio** en el que se encuentra relacionadas a la ubicación de metas. Las pistas están dadas por una **función heurística** denotada por *h(n)*, donde *h(n)* = costo estimado del **camino más barato** desde el **estado** en el nodo *n* hacia un **estado meta.**

2.2. Búsqueda voraz por el mejor primero o *Greedy best-first search*

Es una forma de búsqueda por el mejor primero que consiste en expandir primero el nodo con el valor *h(n)* más bajo —dicho de otra forma, aquel nodo que parecería estar más cerca de la meta— bajo el supuesto de que, al hacer esto, probablemente lleguemos a una solución rápidamente, haciendo uso de la función de evaluación

$f(n) = h(n)$

Tomando el ejemplo del libro de la ruta rumana, suponiendo que se quiere partir desde Arad y llegar a Bucharest:

Figura 3.1. *Un mapa de carreteras de parte de Rumanía, donde las distancias están expresadas en millas.*

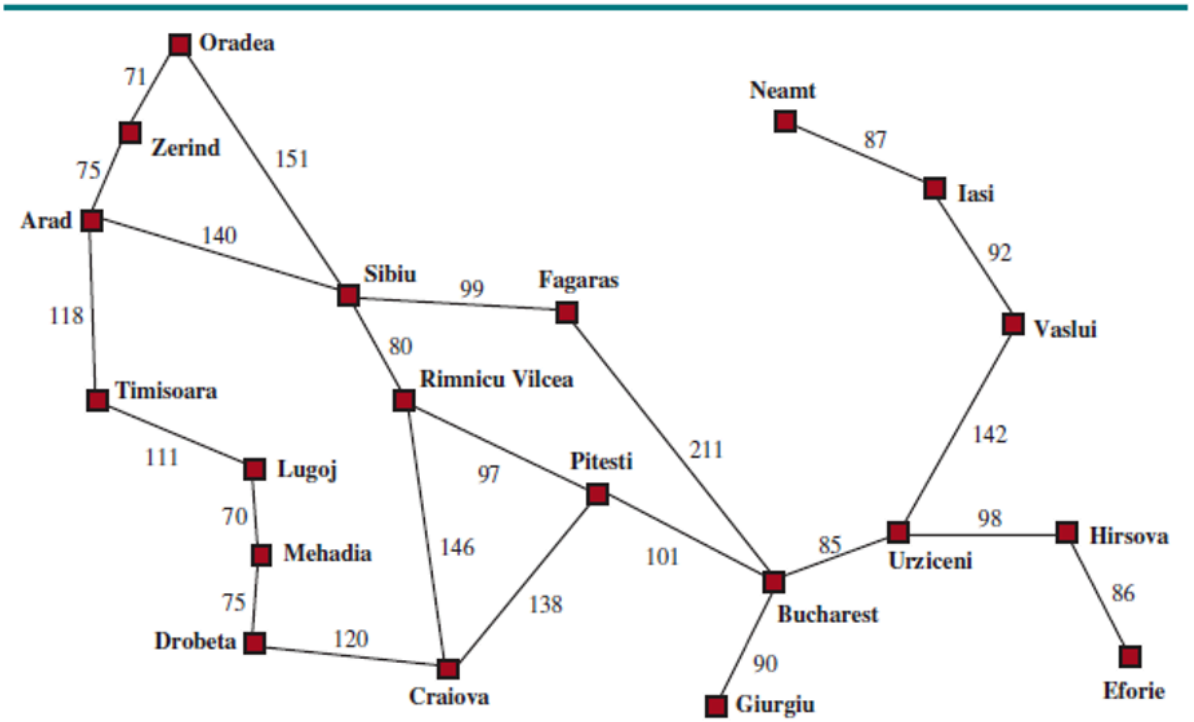


Figure 3.1 A simplified road map of part of Romania, with road distances in miles.

Fuente: Russell, S. J., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach. Pearson.

y haciendo uso de la heurística de la distancia en línea recta, donde la distancia en línea recta de Arad a Bucharest es de 366, se hacen las siguientes evaluaciones:

Figura 3.16.
Los valores de h_{SLD}— las distancias en línea recta hacia Bucharest.

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

Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.

Fuente: Russell, S. J., & Norvig, P. (2021).
Artificial Intelligence: A Modern Approach. Pearson.

El primer nodo a ser expandido desde Arad será Sibiu, debido a que es aquel nodo adyacente que más lejos está (numéricamente) de h_{SLD} = 366.

- f(Sibiu) = 253
- f(Zerind) = 374
- f(Timișoara) = 329

Luego, desde Sibiu, el siguiente a ser expandido será Fagaras por el criterio anterior.

- f(Fagaras) = 176
- f(Rimnicu Vilcea) = 193

La solución encontrada no siempre será la mejor, debido a la naturaleza voraz del algoritmo; en cada iteración busca estar lo más cerca posible de una meta, lo que puede afectar el resultado de no tener cuidado.

2.3. Búsqueda A* o *A* Search*

Es el algoritmo de búsqueda informada más común —pronunciado búsqueda A estrella—, el cual hace uso de la función de evaluación:

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

donde $g(n)$ es el costo del camino del **estado inicial** hasta el nodo *n,* y $h(n)$ es el costo *estimado* del camino más corto desde *n* a un **estado meta**, de tal forma que $f(n)$ = costo estimado del mejor camino que sigue desde *n* hasta una meta.

Siguiendo el anterior ejemplo, se tendrá que $g(n)$ viene a ser la distancia desde el punto inicial (Arad) hasta otra ciudad (nodo *n*).

De esta forma, al evaluar con los distintos nodos:

- f(Sibiu) = g(Sibiu) —distancia relativa de Arad a Sibiu— + h(Sibiu) —calculado anteriormente, 253 millas según figura 3.16—: 393 = 140 + 253
- f(Timisoara): 447 = 118 + 329
- f(Zerind): 449 = 75 + 374

Se tiene que primero se abre Sibiu; sin embargo, luego de evaluar los nodos de Sibiu, se tiene que:

- f(Arad): 280 (ida y vuelta) + 366 = 646
- f(Fagaras): 415 + 176 = 591
- f(Oradea): 291 + 380 = 671
- f(Rimnicu Vilcea): 220 + 193 = 413

El nodo por el cual se ha de seguir buscando es Rimnicu Vilcea. Al final, el camino termina siendo Arad -> Sibiu -> Rimnicu Vilcea -> Pitesti -> Bucharest, con una distancia de 418 millas.

2.4. Búsqueda A* con pesos o *Weighted A* Search*

Similar a la búsqueda A*, en este caso, se hace uso de un multiplicador junto con la heurística; de esta forma, la función de evaluación toma la forma:
$f(n) = g(n) + W \times h(n)$ donde W toma un valor mayor a 1.

A diferencia de la búsqueda A* que, dada una heurística admisible, siempre optimiza el costo, el motivo principal por el cual uno usaría la búsqueda A* con pesos sería para explorar menos nodos, lo que ahorraría tiempo y espacio. Dicho de otra manera, si es posible conformarse con soluciones **satisfactorias.**

Siguiendo el ejemplo anterior, supongamos que ahora disponemos del índice de desvío, un índice que manejan los ingenieros de cambios para estimar el mejor camino entre 2 puntos; se hace esto para tomar en cuenta la curvatura típica de los caminos. Por ejemplo, si hay una distancia entre 2 ciudades de 10 millas, una buena estimación sobre la distancia del mejor camino entre ellos es de 13 millas.

Dado que Rumanía es un país relativamente pequeño, supongamos que el índice de desvío es de 1.2.

Tomando en cuenta esto, el camino resultante es de
f(Sibiu): 140 + 1.2 * 253 = 443.6
f(Timisoara): 118 + 1.2 * 329 = 512.8
f(Zerind): 75 + 1.2 * 374 = 523.8

luego

f(Arad): 280 + 1.2 * 366 = 719.2 (no se toma porque sería volver)
f(Fagaras): 239 + 1.2 * 176 = 450.2
f(Oradea): 291 + 1.2 * 380 = 747
f(Rimnicu Vilcea): 220 + 1.2 * 193 = 451.6

en Fagaras:

f(Sibiu): 338 + 1.2 * 253 = 641.6
f(Bucharest): 450 + 1.2 * 0 = 450

Por ello, según este algoritmo, el camino a tomar sería Arad -> Sibiu -> Fagaras -> Bucharest.

3. Algoritmos de búsqueda local

Estos algoritmos funcionan buscando desde un estado inicial a estados vecinos, sin ir registrando los caminos ni el conjunto de estados que han sido alcanzados. Eso significa que **no son sistemáticos**—podrían nunca llegar a explorar una parte del espacio de búsqueda donde se pueda encontrar una solución. Sin embargo, usan poca memoria y frecuentemente encuentran soluciones razonables en espacios grandes o infinitos, donde sería inviable hacer uso de algoritmos sistemáticos.

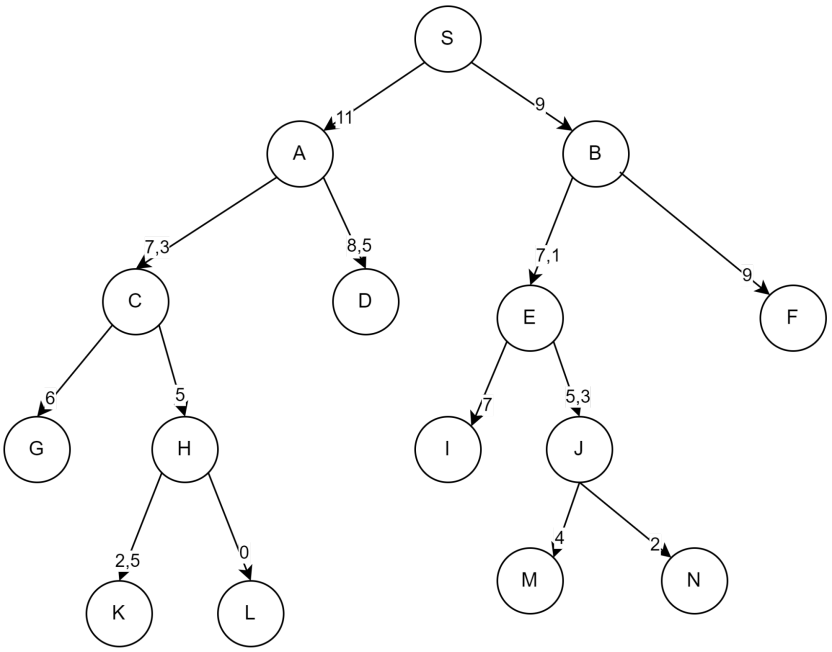
3.1. Haz local o *Local beam search*

A diferencia de otras búsquedas como la escalada simple o el descenso de gradiente que siempre avanzan hacia el mejor valor cercano —el cual puede hacer que no se llegue necesariamente a la solución óptima global—, y solo se registra un estado a la vez, la búsqueda de haz local toma en cuenta *k* estados en vez de solo uno. Se empieza con un número *k* de estados generados y, en cada paso, se generan todos los sucesores de todos los estados *k*. Si alguno de esos sucesores es un estado meta, el algoritmo se detiene; de lo contrario selecciona los mejores *k* sucesores de la lista que fue generada y realiza otra iteración.

Suponga que tiene el siguiente grafo, el cual supone estados cuyos valores representan un costo y la meta es encontrar el nodo que más se acerque a la heurística $h = 0$ (es decir, la solución que se busca es aquella donde el costo sea 0).

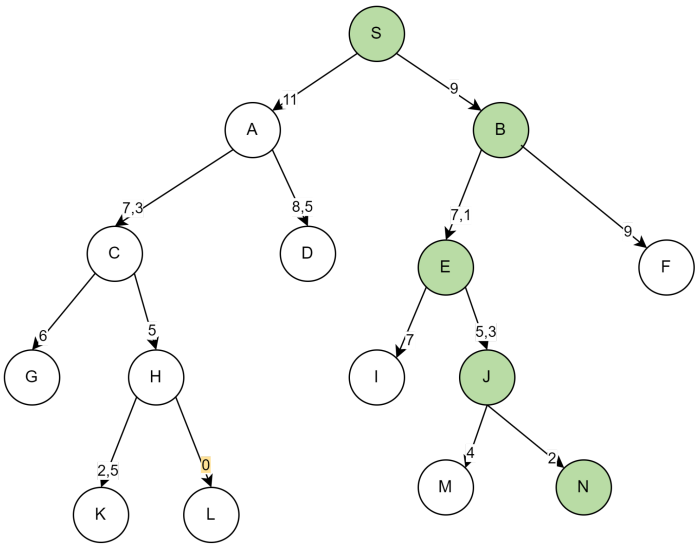
Figura 4.

Representación en forma de grafo de problema de haz local planteado



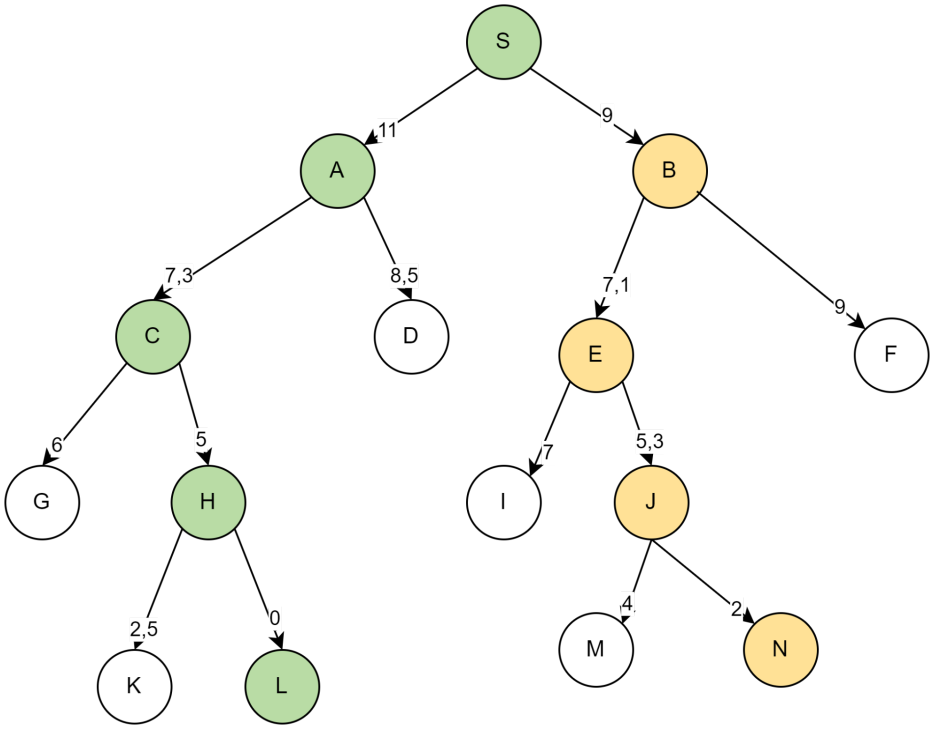
Fuente: Elaboración propia.

Si se fuera a utilizar el algoritmo de descenso de gradiente, tendríamos el siguiente resultado:



Se puede observar que, si bien se llegó a una solución bastante aproximada, el querer restringir la memoria utilizada en pos de economizar recursos no siempre nos brinda la mejor solución posible.

El algoritmo se detuvo en el estado *N* que tiene un costo de 2, ignorando completamente el nodo que realmente se acerca más al valor *h* de 0, el cual es el estado *L*.
Al hacer uso del algoritmo de haz local, con un valor de $k = 2$, se tiene la siguiente forma:



A pesar de que se recorrieron más estados —9 estados visitados frente a 5 con la búsqueda del descenso de gradiente—, esta vez sí fue posible encontrar el nodo *n* cuyo valor se aproxima más a la heurística. Parece similar a la búsqueda de los reinicios en aleatorio, pero no; mientras que cada proceso de búsqueda en la búsqueda de reinicios en aleatorio sucede independientemente de cada uno, la búsqueda de haz local corre estos procesos de forma paralela.

Referencias

Russell, S. J., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach.* Pearson.

Sharma, S. (2020). *Local Beam Search Algorithm in Artificial Intelligence.* YouTube. https://youtu.be/Ad29SjJ1GwA