

**Tema No. 2:** Métodos de solución de problemas.

**Actividad No. 21**

**Tipo de clase:** Conferencia.

**Título:** Métodos informados heurísticamente.

### Sumario:

- ✚ Búsqueda primero el mejor.
- ✚ Algoritmo A\*.
- ✚ Algoritmo RBFS.

### Objetivos:

- ✚ Conocer el significado y empleo de la heurística.
- ✚ Conocer métodos de búsqueda heurística más usados.
- ✚ Valorar los diferentes métodos de búsqueda heurísticas.

### Introducción:

Los métodos de búsqueda en anchura y en profundidad son denominados métodos ciegos de búsqueda. En general, serán muy ineficaces y sin aplicación práctica debido al crecimiento exponencial del tiempo y del espacio que requieren.

La eficiencia en la búsqueda puede mejorar en gran medida si existe una forma de ordenar las selecciones de modo que las más prometedoras se exploren primero. En muchas situaciones, usted puede hacer mediciones para determinar un ordenamiento razonable. En esta conferencia usted podrá aprender algunos de los métodos de búsquedas que sacan provecho de tales mediciones, y que se conocen como métodos informados heurísticamente.

Los métodos de búsqueda heurística disponen de alguna información sobre la proximidad de cada estado a un estado objetivo, lo que permite explorar en primer lugar los caminos más prometedores.

### Búsqueda Heurística.

Los métodos de búsqueda heurísticas están orientados a reducir la cantidad de búsqueda requerida para encontrar una solución. Cuando un problema es presentado como un árbol de búsqueda el enfoque heurístico intenta reducir el tamaño del árbol cortando nodos pocos prometedores. Estos métodos se llaman métodos fuertes porque ellos son más poderosos que los estudiados hasta aquí al incorporar conocimiento heurístico o heurística. Hay una contradicción entre generalidad y potencia en el sentido que los métodos débiles son esencialmente aplicables universalmente mientras que los fuertes son menos universales en su aplicabilidad y el conocimiento o heurística usado en un problema dado puede no ser totalmente aplicable o ser inaplicable en otro dominio o tarea.

En esencia una heurística es simplemente un conjunto de reglas que evalúan la posibilidad de que una búsqueda va en la dirección correcta. Generalmente los métodos de búsqueda heurísticas se basan en maximizar o minimizar algunos aspectos del problema. Un ejemplo sencillo de heurística es el siguiente:

NORTE: vegetación verde y movimiento de animales

SUR: vegetación amarilla

ESTE: vegetación amarilla

OESTE: vegetación verde

Evidentemente la vegetación verde es un indicio de que hay humedad, luego es muy probable que exista agua en la superficie o subterránea. El movimiento de animales puede indicar que ellos se dirigen allí a beber, lo cual sugiere que el agua está en la superficie. Esta información le dice al hombre que debe dirigirse al norte, constituye una heurística.

La Heurística no garantiza que siempre se tome la dirección de la búsqueda correcta, por eso este enfoque no es óptimo sino suficientemente bueno. Frecuentemente son mejores los métodos heurísticos que los métodos de búsquedas a ciegas. Las desventajas y limitaciones principales de la heurística son:

- La flexibilidad inherente de los métodos heurísticos pueden conducir a errores o a manipulaciones fraudulentas.
- Ciertas heurísticas se pueden contradecir al aplicarse al mismo problema, lo cual genera confusión y hacen perder credibilidad a los métodos heurísticos.
- Soluciones óptimas no son identificadas. Las mejoras locales determinadas por la heurística pueden cortar el camino a soluciones mejores por la falta de una perspectiva global. La brecha entre la solución óptima y una generada por heurística puede ser grande.

### **Funciones de evaluación heurística.**

Una función de evaluación heurística es una función que hace corresponder situaciones del problema con números. Es decir, da una medida conceptual de la distancia entre un estado dado y el estado objetivo. Estos valores son usados para determinar cuál operación ejecutar a continuación, típicamente seleccionando la operación que conduce a la situación con máxima o mínima evaluación.

Algunas consideraciones sobre las funciones heurísticas son:

- a) La función debe dar un estimado útil y realista del mérito de un estado particular.
- b) La evaluación de la función en general no debe requerir un gran cálculo en su aplicación. Si la evaluación de la función es computacionalmente compleja puede ser más eficiente hacer una búsqueda a ciegas en lugar de gastar recurso (tiempo y memoria) en el cálculo de la función.
- c) Frecuentemente el costo de una solución exacta a un problema flexibilizado es una buena heurística para el problema original. Un problema flexibilizado es uno obtenido a partir del problema original simplificando las restricciones sobre los operadores. La idea es que el número exacto de movimientos requeridos para resolver un problema más simple puede ser fácil de calcular y puede servir como un estimado de la cantidad de movimientos necesarios para resolver el problema original.
- d) Es siempre mejor usar una función heurística con valores más altos que otras, siempre que esta no esté sobreestimada. Para un problema puede haber una colección de heurísticas admisibles  $h_1, \dots, h_m$ . Si una de ellas domina a las otras, es decir alcanza valores mayores para todos los nodos, se debe seleccionar esta. Si ninguna es dominante lo mejor es definir una heurística compuesta de la forma siguiente:  
$$h(n) = \max(h_1(n), \dots, h_m(n)).$$

De esta forma  $h$  dominará todas las heurísticas individuales.
- e) Otra forma de inventar una buena heurística es usar información estadística. Esto puede ser hecho realizando una búsqueda sobre una cantidad de problemas de entrenamiento, por ejemplo, en el juego de las ocho piezas cien configuraciones generadas aleatoriamente.

- f) Frecuentemente es posible seleccionar rasgos de un estado que contribuyen a su evaluación heurística. La función de evaluación heurística puede ser construida como una combinación lineal de estos rasgos. Los rasgos pueden tener un peso que indique su importancia.
- g) Otro tipo de modelo flexibilizado para un problema son los modelos analógicos. Aquí el modelo auxiliar flexibilizado extrae su potencia no de simplificar la estructura del problema a resolver sino de usar procesos de búsqueda que fueron empleados con éxito en problemas análogos.

#### Ejemplos de funciones heurísticas

- a) El problema de las ocho reinas

Asumiendo la variante incremental del juego, es decir poner las reinas una a una de modo que no se ataquen mutuamente, supóngase que se tienen situadas tres reinas y tenemos que decidir dónde colocar la cuarta. El papel de la heurística aquí sería ofrecer un criterio para decidir cuál de las tres posiciones indicadas es la más promisorio de conducir a una solución satisfactoria.

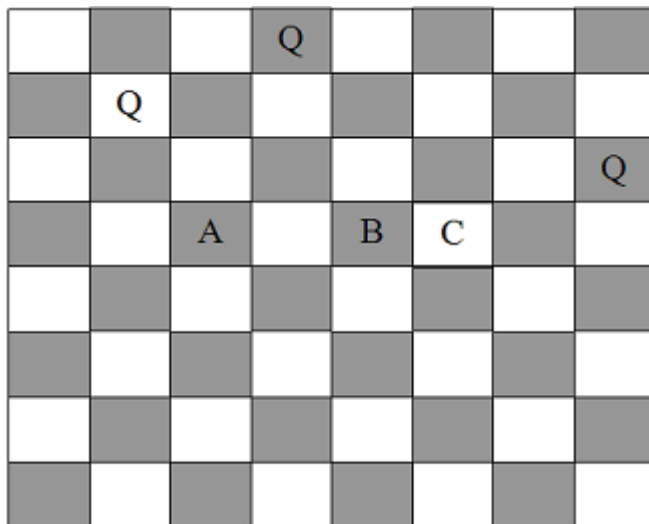


Figura 1: Ejemplo de situación del juego de las 8 reinas.

Al deducir una heurística para este problema podemos razonar que para poder colocar las ocho reinas tenemos que dejar libres la mayor cantidad de opciones como sea posible para futuras adiciones de reinas.

Esto significa determinar la cantidad de casillas en las filas no utilizadas que quedarían no atacadas al colocar la cuarta reina en A, B o C. Una casilla candidata será preferida si deja la mayor cantidad de casillas no atacadas en el resto del tablero. Consecuentemente el número de casillas no atacadas constituye una medida de su mérito, es decir, es la función de evaluación heurística para este problema.

Al calcular la función heurística para las posiciones A, B y C se tiene:

$$f(A) = 8$$

$$f(B) = 9$$

$$f(C) = 10$$

Otra alternativa o enfoque heurístico es el siguiente. Las filas no usadas no tienen igual estatus ya que aquellas con pocas casillas no atacadas tienden a quedar bloqueadas más rápidamente que filas con muchas casillas no atacadas. Consecuentemente, si queremos minimizar la posibilidad de un futuro bloqueo debemos focalizar nuestra atención sobre la fila con menor número de casillas

no atacadas, sea  $f'$  la heurística que me devuelve la cantidad de casillas atacadas de la fila con menor número de casillas atacadas. Para esta función tenemos,

$$f'(A) = 1$$

$$f'(B) = 1$$

$$f'(C) = 2$$

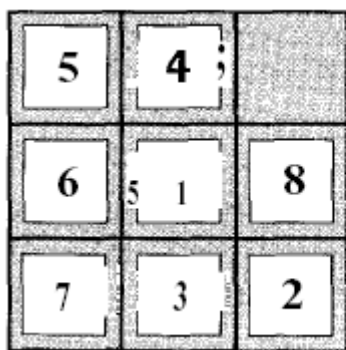
Nótese que con esta heurística C también es la alternativa preferida. Además, si alguna opción candidata toma valor cero para  $f$  o  $f'$  no tiene sentido considerarla pues eventualmente será un nodo muerto, la función  $f'$  supera a  $f$  en que esta detecta todos los nodos muertos predichos por  $f$  y muchos más.

b) El problema de las ocho piezas.

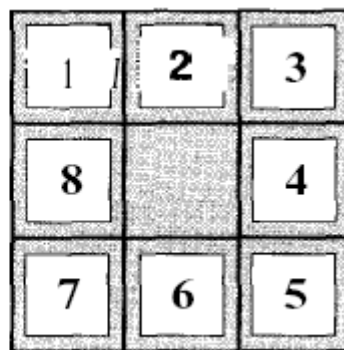
El objetivo de este juego es reordenar una configuración inicial dada de ocho piezas sobre un tablero de tres por tres en una configuración final.

En el reordenamiento sólo se permite desplazar fichas a la casilla vacía, siempre que sea adyacente, es decir la regla es:

Una pieza puede moverse de la posición X a la Y si la posición X es adyacente a Y pero además la posición Y está vacía.



Estado actual



Estado objetivo

Figura 2: El problema de las ocho piezas.

Una solución típica para este problema requiere alrededor de veinte pasos, aunque este número varía dependiendo del estado inicial. El factor de ramificación es aproximadamente tres (cuando la casilla vacía está en el centro hay cuatro movimientos posibles y cuando está en una esquina hay dos). Esto significa que una búsqueda exhaustiva a profundidad veinte podría requerir cerca de  $3^{20} = 3.5 \times 10^9$  estados. Eliminando los estados repetidos quedarían  $9! = 362\,880$  ordenamientos diferentes. Esta es aún una cantidad grande de estados por lo que se requiere encontrar una buena función heurística.

Un razonamiento a seguir para construir la función heurística para este problema es estimar cuán cerca un estado está del objetivo. Hay dos variantes comúnmente usadas para estimar la proximidad de un estado a otro:

- Cantidad de piezas mal ubicadas, aquellas por las cuales los dos estados difieren.
- La suma de las distancias (horizontal y vertical) de las piezas a sus posiciones en el estado objetivo.

Al aplicar estas heurísticas al estado actual de la figura 2 tenemos que:

$$h1(EA) = 7$$

$$h2(EA) = 2 + 3 + 3 + 2 + 4 + 2 + 0 + 2 = 18$$

La variante de encontrar funciones heurísticas flexibilizando el modelo del problema se puede ilustrar con este juego. A partir de la regla que gobierna el juego se pueden generar tres problemas flexibilizados removiendo una o más de las condiciones:

- (a) Una ficha se puede mover de X a Y si X está adyacente a Y (elimina la condición que Y esté vacío)
- (b) Una ficha se puede mover de X a Y si Y está vacía (elimina la condición de que ambas sean adyacentes)
- (c) Una ficha se puede mover de X a Y (elimina ambas condiciones)

La cantidad de movimientos requeridos para resolver el problema (a) es exactamente igual a la distancia Manhattan.

La cantidad de movimientos requeridos para resolver el problema (C) es exactamente igual a la cantidad de fichas fuera de lugar, es decir que están en una posición diferente a la que deben tener en el estado objetivo; la cual coincide con  $h_1$ .

La función  $h_1$  es más barata pero menos precisa que  $h_2$ .

En el caso del problema (b) la cantidad de movimientos requeridos para resolverlo es la cantidad de veces que la posición vacía tiene que ser intercambiada con otra ficha, lo cual sugiere otro estimado heurístico para el problema original.

c) El problema de las carreteras.

Dado un mapa con las carreteras existentes entre ciudades, queremos encontrar el camino más corto entre las ciudades A y B conociendo las longitudes de las carreteras entre las ciudades. Estas distancias las podemos colocar en una tabla de modo que  $d(i,j)$  indica la distancia por carretera entre las ciudades i y j. ¿Qué información le da el mapa a un hombre que no le ofrece esta tabla de distancias?

El hombre hace un estimado de la distancia euclidiana en el mapa entre dos ciudades, por ejemplo, si la distancia aérea de D a B es más corta que la existente entre C y B. La ciudad D parece un mejor candidato que C. En ausencia del mapa podíamos suplantar esta información visual por una tabla adicional con la distancia aérea entre las ciudades. Esta tabla  $h(i)$  contiene la distancia aérea de la ciudad i al objetivo B.

Esta sería la función heurística:  $d(A,i) + h(i)$ .

### **Búsqueda primero el mejor (best first)**

#### **Características.**

- Se tiene un control de todos los estados terminales y sus heurísticas.
- Se toma como raíz el estado de valor óptimo entre todos los estados terminales.
- Se generan en amplitud todos los estados hijos del óptimo (expandir el nodo más idóneo).

La idea del best-first es utilizar una función de evaluación para cada nodo (estimación de la idoneidad de expandir ese nodo).

Aplicación: Expandir siempre el nodo no expandido más idóneo.

### **PASOS DEL BEST-FIRST**

- 1.- A partir del estado inicial genera los correspondientes nodos hijos.
- 2.- Aplica la función heurística que se tenga, a cada uno de los hijos.
- 3.- Selecciona el mejor de los hijos y lo toma como el siguiente estado en la búsqueda del estado meta.
- 4.- Genera los estados hijos de este nuevo estado.
- 5.- Evalúa la función heurística de los hijos antes generados.
- 6.- Selecciona como siguiente estado al cual pasar, al mejor de todos los nodos hijos generados hasta el momento y que no hayan sido evaluados (expandidos).
- 7.- Seguir repitiendo los pasos 4, 5 y 6 hasta llegar al estado final.

Para su operación, el algoritmo necesita dos listas de nodos y una función heurística que estime los méritos de cada nodo que se genere:

1. **ABIERTOS** - Es una variable que contiene los nodos que han sido generados. La función heurística ha sido aplicada a ellos, pero todavía no han sido examinados, es decir no se han generado sus sucesores. **ABIERTOS** puede considerarse como una **COLA DE PRIORIDADES** en la que los elementos con mayor prioridad son los que tienen los valores más prometedores, dados por la función heurística.
2. **CERRADOS** - Es una variable que contiene los nodos que han sido examinados. Es necesario tener esta información, para que la búsqueda sea en un grafo y no en un árbol.
3. **FUNCIÓN HEURÍSTICA** - Permite que el algoritmo busque primero por senderos que son o parecen más prometedores.

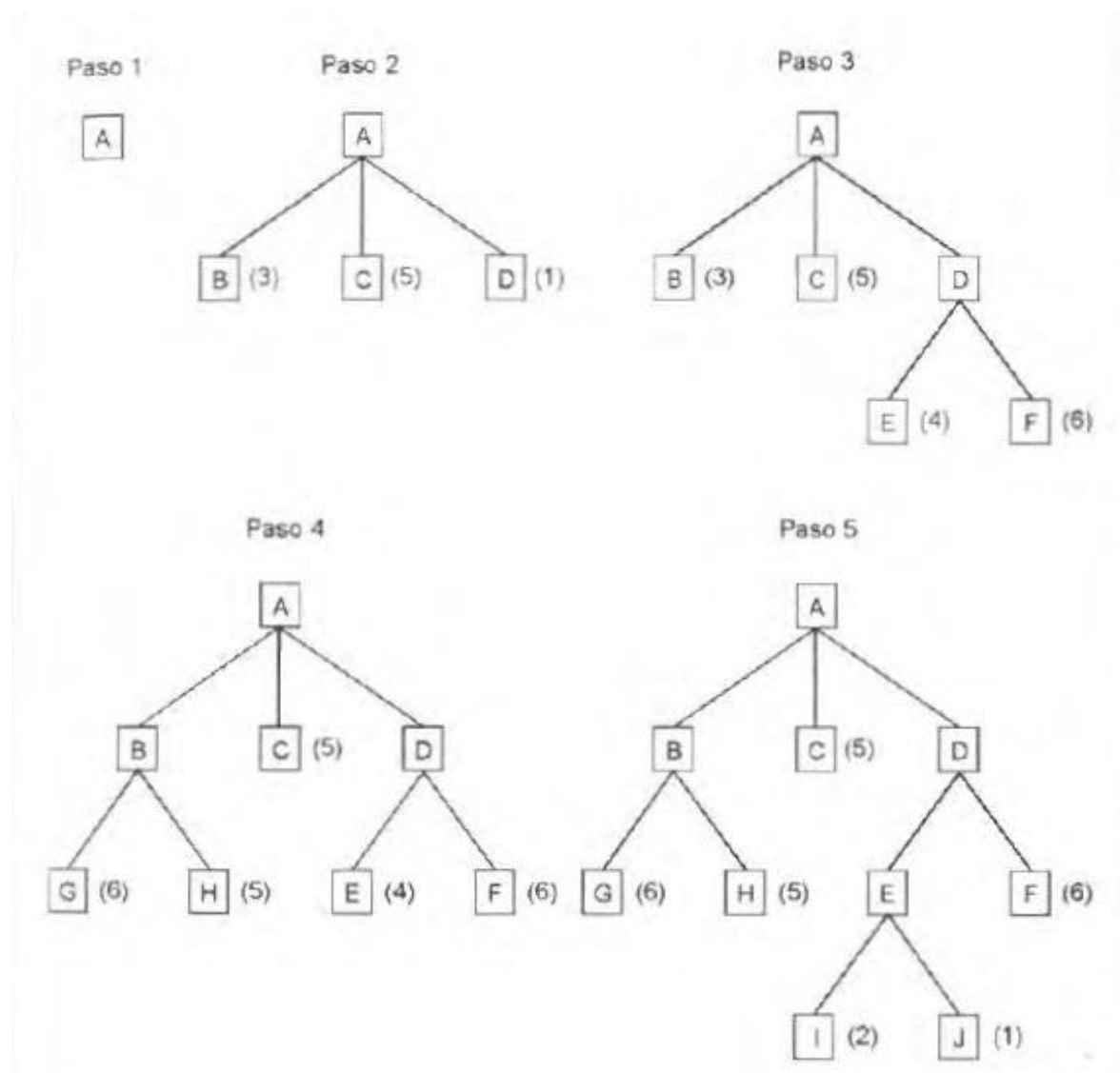


Figura 4. Un ejemplo de la búsqueda del primero el mejor



### Algoritmo A\*

Para muchas aplicaciones, es conveniente definir esta función  $f'$ , como la suma de dos, que se las llamará  $g$  y  $h'$ . La función  $g$  es una medida del costo de llegar desde el nodo inicial al nodo actual. La función  $h'$  es una estimación del costo adicional para llegar desde el nodo actual al estado objetivo. Aquí es donde se explota el conocimiento que se dispone sobre el dominio del problema.

Es decir, la función combinada  $f'$  representa una estimación del costo de llegar desde el estado inicial hasta el estado objetivo, siguiendo el sendero que ha generado el nodo actual. Si el nodo actual ha generado más de un sendero, el algoritmo deberá dejar registrado sólo el mejor.

**Función de evaluación:  $f(n) = g(n) + h(n)$**  (se denomina **potencia heurística**)

**$g(n)$ :** costo para llegar al nodo  $n$  (lo que he recorrido).

**$h(n)$ :** costo *estimado* para llegar a un nodo solución desde el nodo  $n$  (estimado de lo que falta por recorrer).

**$f(n)$ :** costo total *estimado* del camino para llegar al objetivo a través del nodo  $n$ .

La búsqueda A\* utiliza heurísticos **admisibles** ( $h^*(n)$ ):

$h^*(n) = h'(n)$  donde  $h'(n)$  es el heurístico perfecto (costo **real** desde el nodo  $n$  al nodo solución).

### Ejemplo 1:

En el problema del viaje por Rumanía:  $hDLR(n)$  es un heurístico admisible (nunca sobreestima la distancia real por carretera a Bucharest).

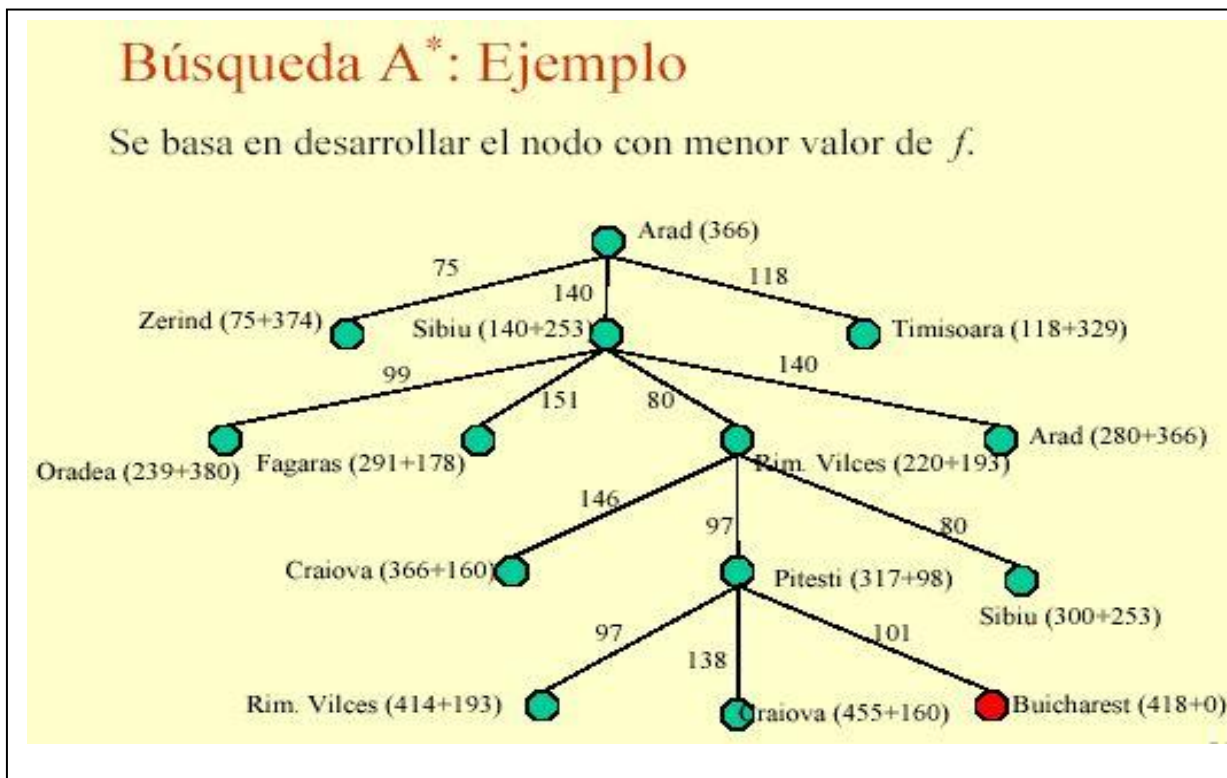


Figura 5: El problema del viaje por Rumania.

### Ejemplo 2:

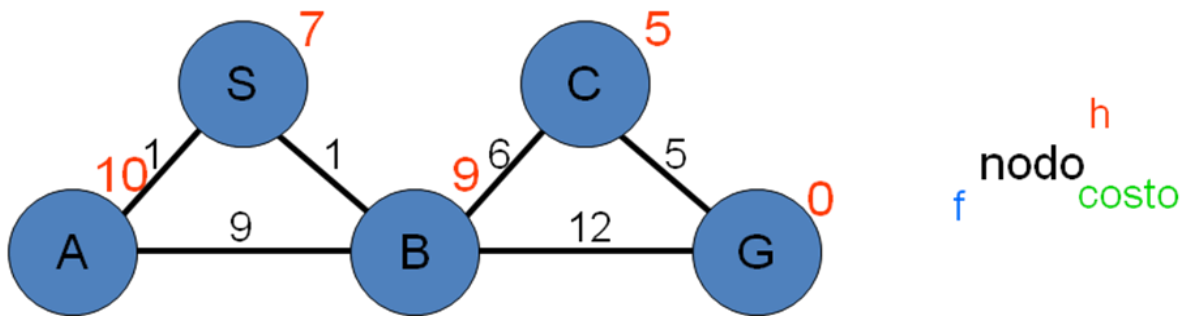


Figura 6: Grafo para recorrido del A\*.

En este grafo S sería el nodo de partida y G el nodo meta. El algoritmo A\* iría recorriendo los nodos del grafo según se muestra en la figura 7

Primero los vecinos de S que serían A y B, a los cuales se les calcula el valor de su función f, la cual da valores de 11 y 10 respectivamente, lo que nos dice que el siguiente nodo a ser expandido sería el B.

Al expandir B se obtienen A, C y G, a con sus respectivos valores de f, 20, 12 y 13.

Notar que estos nodos representan trayectorias, para el primer valor de A que se obtuvo fue llegando directamente de S, mientras que la A que se acaba de obtener recorre el camino a través de B.

Tengan en cuenta además que aunque se haya generado el nodo G, este tiene valor de la función f mayor que otro nodo, por tanto no le corresponde ser expandido, es solo en ese instante que se detiene la búsqueda.

El algoritmo continúa de esta forma expandiendo el nodo de menor valor de f, ubicado a la cabeza de la cola de prioridad, hasta que le corresponde analizar a G, habiendo pasado por el camino SBC anteriormente, y como es un estado meta se detiene.

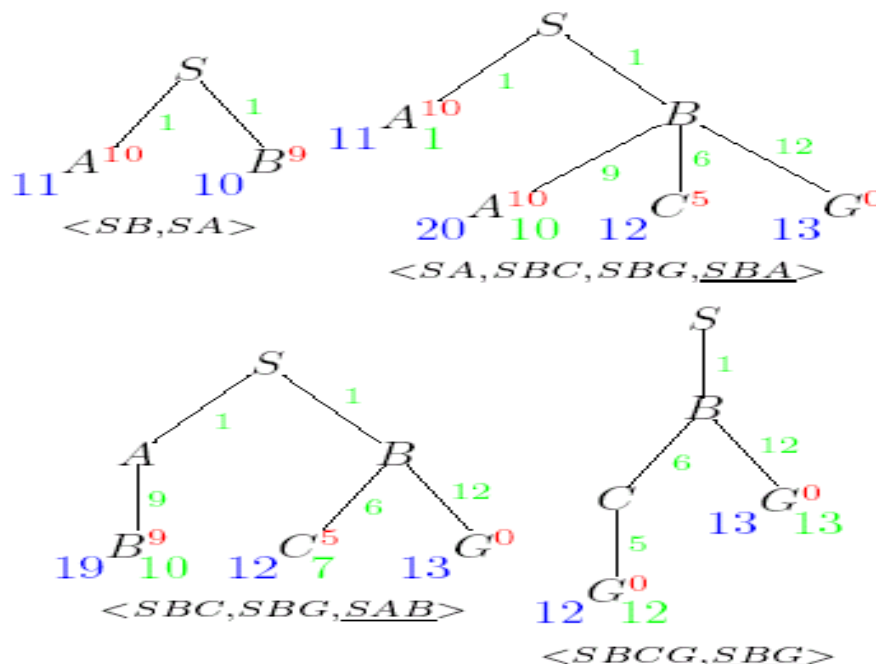
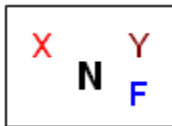




Figura 7: Exploración del A\*.

A continuación la corrida del algoritmo A\* utilizando otra representación gráfica.



**X**, Representa en el número que se generó,

**Y**, en el número que expandió,

**F**, Valor de la función “costo + heurística” y,

**N**, representa el nodo.

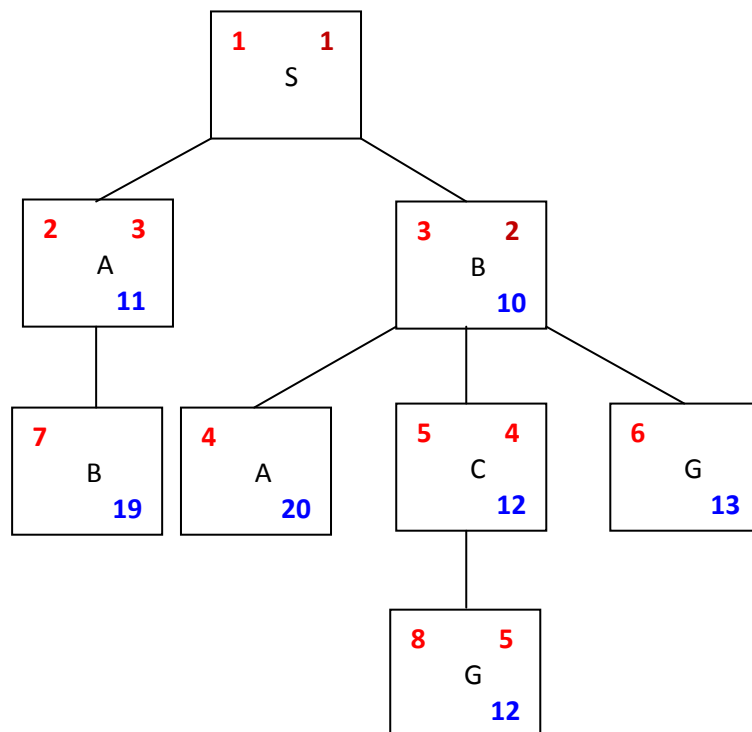


Figura 8: Otra manera de hacer la exploración del A\*.

**Nota:** Tenga en cuenta que se había llegado al estado meta inicialmente con un valor de 13. Aquí no se termina el algoritmo porque hay estados con menor valor que no se han expandido.

Usualmente, el costo de ir de un nodo a su sucesor,  $g$ , se fija en una constante igual a 1, cuando se desea encontrar un sendero a la solución, que involucre el menor número de pasos. Si por el contrario nos interesa encontrar el camino menos costoso y algunos operadores cuestan más que otros, se asume un valor de  $g$ , que refleje esos costos. Un valor de  $g$  igual a cero significaría que simplemente nos interesa llegar a alguna solución, de cualquier manera.

Si  $h'$  es un estimador perfecto de  $h$ , hará que A\* converja inmediatamente al objetivo, sin búsqueda. Mientras mejor sea  $h'$ , más cerca se estará de alcanzar esta aproximación directa.

Si  $h'$  vale cero, la búsqueda será controlada por  $g$ . Si el valor de  $g$  es también cero, hará que la búsqueda sea aleatoria. Si el valor de  $g$  es siempre 1, hará que la búsqueda sea primero en anchura. Para los casos en que  $h'$  no sea perfecto ni cero, y nunca llega a sobrestimar el valor de  $h$ , el algoritmo  $A^*$  está garantizado que encontrará un sendero óptimo a un objetivo, en caso de que exista solución. Cuando un algoritmo garantiza el encontrar una solución óptima, si esta existe, se dice que es **admisible**.

### Teorema de Admisibilidad

Si  $h'$  nunca sobrestima a  $h$  entonces  $A^*$  es admisible. La única manera de garantizar que  $h'$  nunca sobrestime a  $h$  es haciéndolo cero, pero con ello estamos en la búsqueda primero en anchura, es decir el algoritmo es admisible, pero no eficiente. Pero existe un corolario que puede ser de utilidad en casos prácticos:

**COROLARIO:** Si  $h'$  muy rara vez sobrestima  $h$  por más de  $\delta$  entonces el algoritmo  $A^*$  rara vez encontrará una solución cuyo costo sea mayor más que en  $\delta$  que el costo de la solución óptima.

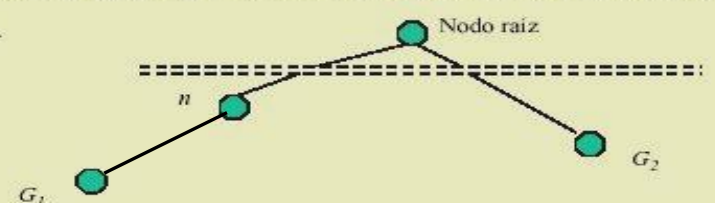
Bajo ciertas condiciones, el algoritmo  $A^*$  puede ser óptimo, en el sentido de generar el menor número de nodos posibles, en el proceso de encontrar una solución, pero bajo otras condiciones puede no serlo.

### Teorema:

La búsqueda  $A^*$  es **óptima**.

**Prueba de que el algoritmo  $A^*$  es óptimo**

Supongamos un nodo objetivo subóptimo  $G_2$  y un nodo  $n$  sin expandir que se encuentre en el camino a la solución óptima  $G_1$ .



$f(G_2) = g(G_2)$  ya que  $h(G_2) = 0$   
 $> g(G_1)$  ya que  $G_2$  es subóptimo  
 $\geq f(n)$  ya que  $h$  es admisible

Dado que  $f(G_2) > f(n)$ ,  $A^*$  nunca seleccionará  $G_2$  para expandir.

### Evaluación de búsqueda $A^*$

1. ¿Completa? Sí.

2. Complejidad:

- a) Temporal: Exponencial en [error relativo de  $h$  x longitud del camino a la solución].
- b) Espacial: Guarda todos los nodos en memoria

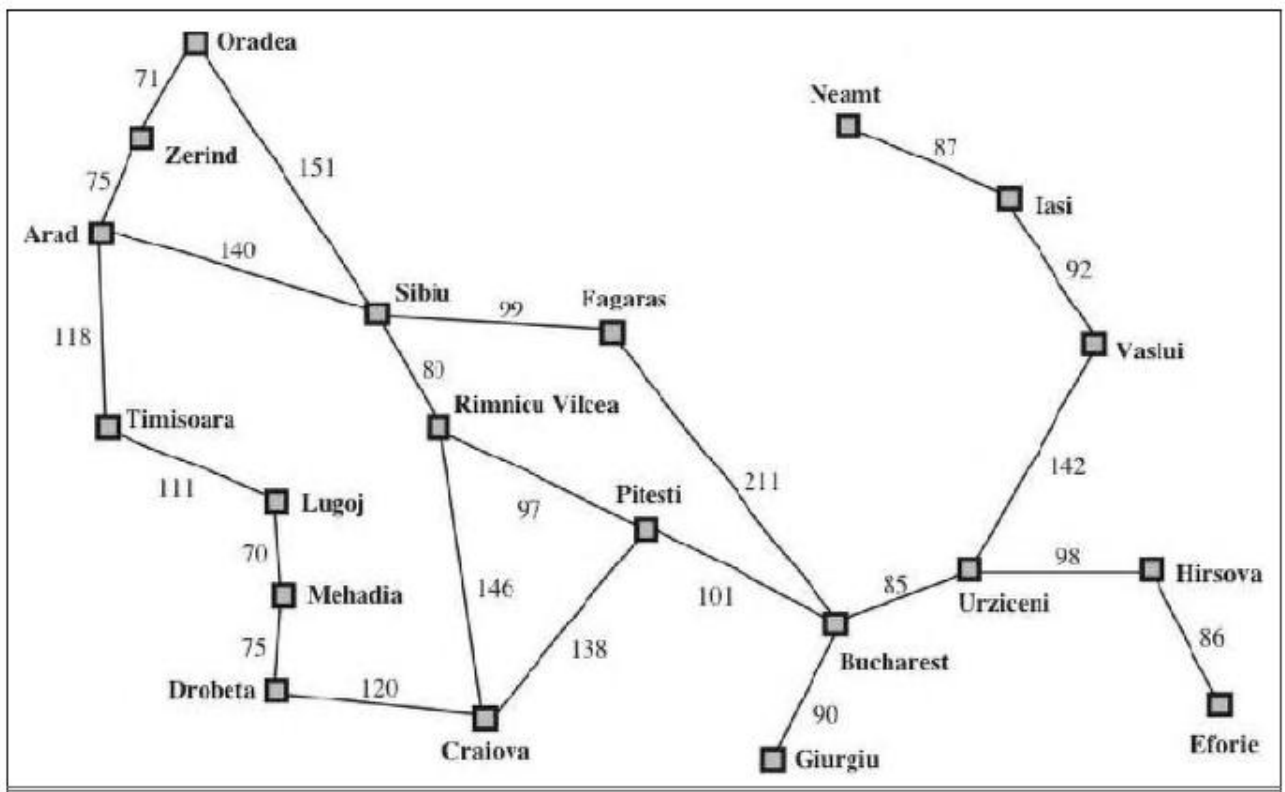
### 3. ¿Óptimo?: Sí.

Algoritmo A\*:

1. Crear una lista con el nodo raíz
2. Hasta que la lista esté vacía o se alcance la meta:
  1. Si el primer elemento es la meta entonces ve al paso 3  
Sino elimina el primero de la lista y agrega sus hijos sumando  $g + h$  (heurística 1)
  2. Ordena los elementos de acuerdo al costo y elimina los repetidos de costo mayor (heurística 2)
3. Expande todos los nodos con costo menor que el nodo meta.

### RBFS

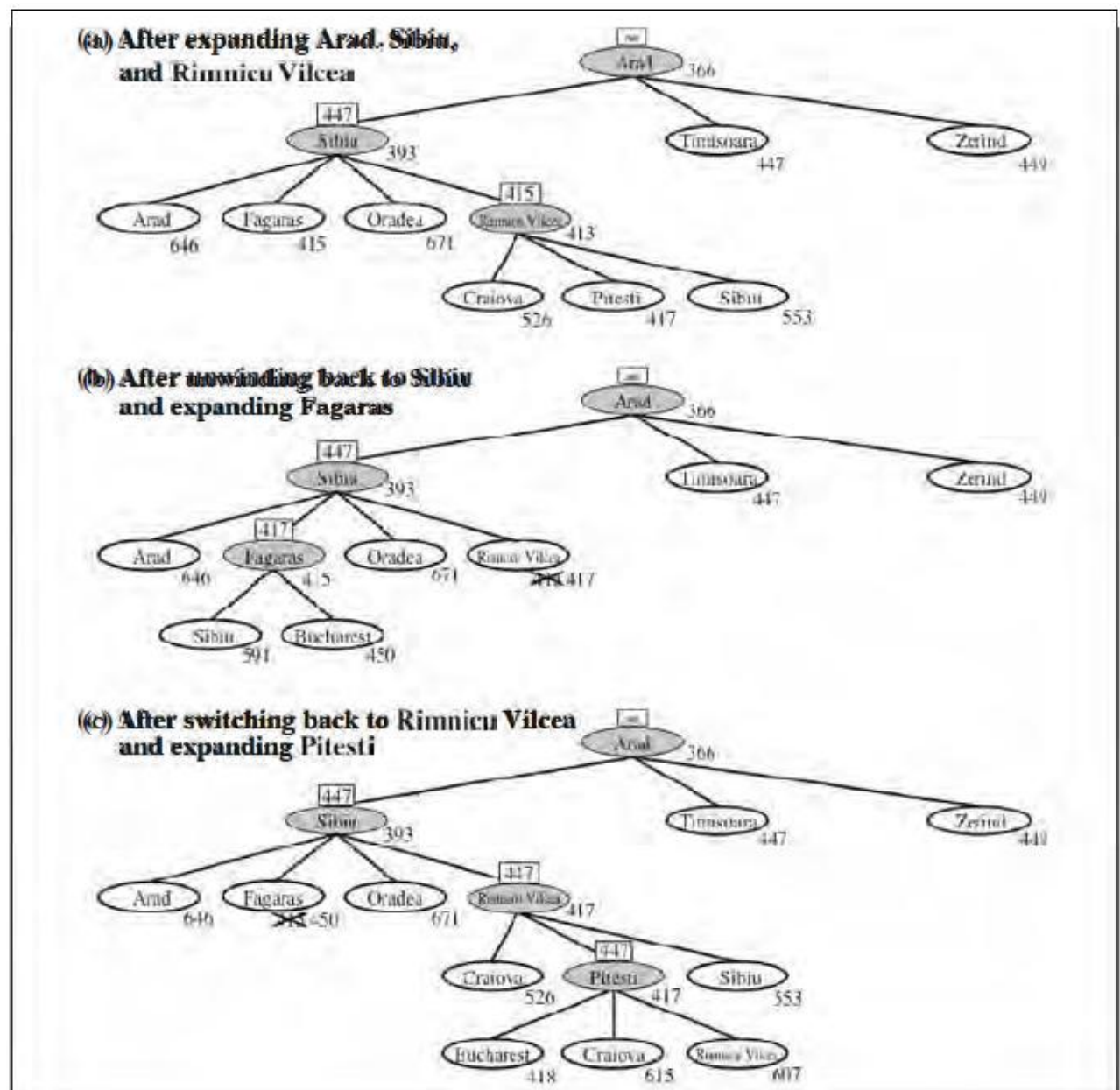
Recursive best-first search (RBFS) es un algoritmo simple recursivo que intenta imitar un best-first standard, pero usando solamente el espacio lineal. Su estructura es similar a la del DFS, pero en vez de continuar indefinidamente a lo largo del camino actual, utiliza la variable f-limit para almacenar el valor del mejor camino alternativo desde cualquier ancestro del nodo actual. Si el nodo actual excede el f-limit, la recursión retorna al camino alternativo y se reemplaza el valor f de cada nodo a lo largo del camino con el mejor valor f de sus hijos. De esta manera, el RBFS recuerda el f-value de la mejor hoja en el subárbol desechado y puede por tanto decidir si es mejor reexpandir ese subárbol en algún momento posterior. A continuación veamos un ejemplo de este recorrido. Supongamos que tenemos el siguiente mapa:



con la siguiente tabla de heurísticas (distancias en línea recta hasta Bucarest):

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

En la siguiente figura se puede apreciar un ejemplo de ejecución del algoritmo:



En la figura, el valor del f-límite de cada llamado recursivo se muestra en el tope de cada nodo actual, en un cuadrado, y cada nodo está etiquetado con su costo f. De acuerdo a a), el camino vía Rimnicu Vilcea es seguido hasta que la mejor hoja actual tiene un valor que es peor que la mejor vía alternativa (Fagaras). En b) se va retornando de la recursividad y el valor de la mejor hoja del subárbol desechado (450) se almacena en backtracking en Rimnicu Vilcea, luego Fagaras es expandido, revelando un valor de mejor hoja de 450 en Bucarest. Ya para finalizar, en c) se retorna de la recursividad y el mejor valor hoja del subárbol desechado se almacena hacia Fagaras, entonces Rimnicu Vilcea es expandido. Esta vez, debido a que la mejor alternativa es de 447 (a través de Timisoara) la expansión continúa hacia Bucarest y como es el estado objetivo, termina la búsqueda.

### **Conclusiones**

Existen diferentes métodos de búsqueda, los métodos heurísticos por lo general son mejores que los métodos a ciegas, cuando se ha elegido correctamente la heurística.

Los métodos de solución de problemas no solo son aplicables a la Inteligencia Artificial, sino a cuestiones prácticas de la vida.

### **Bibliografía**

[1] Patrick Henry Winston, Inteligencia Artificial, 3ra edición, 1992.

[2] Elaine Rich & Kevin Knight, Inteligencia Artificial, 2da edición, 1994.

[3] Julio J. Rubio Garcia. Inteligencia Artificial e Ingeniería del Conocimiento I. BÚSQUEDA, 2004

### **Estudio independiente.**

A partir del problema de las 8 piezas descrito en la conferencia, partir del estado actual de la figura aplicando un algoritmo primero el mejor mediante las siguientes transiciones, en ese orden:

-Mover abajo

-Mover a la derecha

-Mover arriba

-Mover a la izquierda

Defina como heurística la cantidad de piezas fuera de su lugar en el estado objetivo. Termine cuando llegue a tener 2 piezas en su lugar.