UNIVERSIDAD EAFIT

Escuela de Ciencias Aplicadas e Ingeniería

Artificial Intelligence

Lecture03 – Heuristic search & Optimality

Contenido

1. Algoritmos de Búsqueda informada
2. Gredy Best-First
3. A* Search
4. Optimalidad
5. Interactive Best First Search
6. Ejercicio Maze-Algorithm

Agentes de Búsqueda Informada

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)***.

Agentes de Búsqueda Informada

Y entonces… **¿Que es una Heurística?**

“This additional information, helps estimate the cost or distance from a given node … to the goal node.”

“Heuristics are mental shortcuts or rules of thumb that simplify decision making and problem-solving processes. … They rely on intuitive judgment and common sense rather than a systematic and exhaustive analysis of every possible option.”

Agentes de Búsqueda Informada

La idea de los algoritmos de búsqueda informada es simplificar el proceso de análisis del agente y tomar decisiones guiadas con el fin de encontrar el estado objetivo

“En la resolución de problemas, las heurísticas ayudan a crear un modelo simplificado del mundo que facilita la generación de soluciones. Reducen la carga cognitiva al centrarse en los aspectos más relevantes del problema. ”

Agentes de Búsqueda Informada

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

Tipos de Heurísticas

Tipo de Heurística	Definición	Ejemplo práctico (profesional/empresarial)
Disponibilidad	Se basa en la facilidad con la que recordamos ejemplos o información reciente. Se asume que lo más "fácil de recordar" es lo más probable.	Un gerente de riesgos sobreestima la probabilidad de fallas porque recientemente ocurrió un problema visible en otro proyecto.
Representatividad	Se juzga la probabilidad de un evento según qué tan similar parece a un caso típico, ignorando datos estadísticos reales.	Un reclutador asume que un candidato que viste de forma "corporativa" será necesariamente el más competente, sin revisar sus métricas reales de desempeño.
Anclaje y ajuste	Se toma un valor inicial (ancla) y se hacen ajustes a partir de él.	En una negociación de salarios, el primer valor mencionado influye fuertemente en el acuerdo final, incluso si no refleja el valor real del puesto.
Reconocimiento	Se selecciona la opción que se reconoce más fácilmente, bajo la suposición de que lo conocido es mejor o más seguro.	Un consumidor elige una marca de café porque "le suena" más confiable que otras desconocidas, aunque no compare precios ni calidad.
Regla de parada (stopping rule)	Se detiene la búsqueda de información cuando se encuentra una opción "suficientemente buena", sin agotar las alternativas.	Un equipo de desarrollo selecciona la primera herramienta de software que cumple los requisitos básicos sin evaluar opciones potencialmente más eficientes.

Gredy 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.

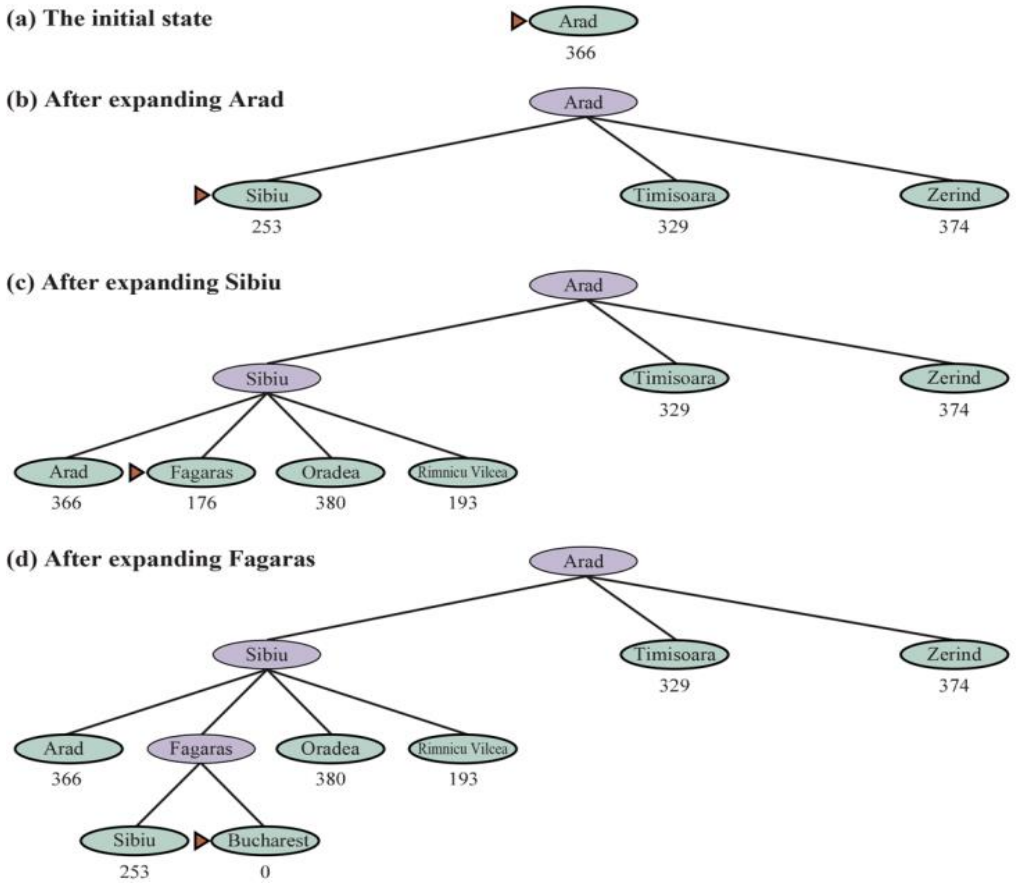
$$f(n) = h(n).$$

Gredy Best-First

Seguiremos con el ejemplo del agente de Rumania, y en este caso, usaremos la heurística de distancia en línea recta, que llamaremos h_{SLD}

	h_{SLD}		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

Gredy Best-First



A* Search

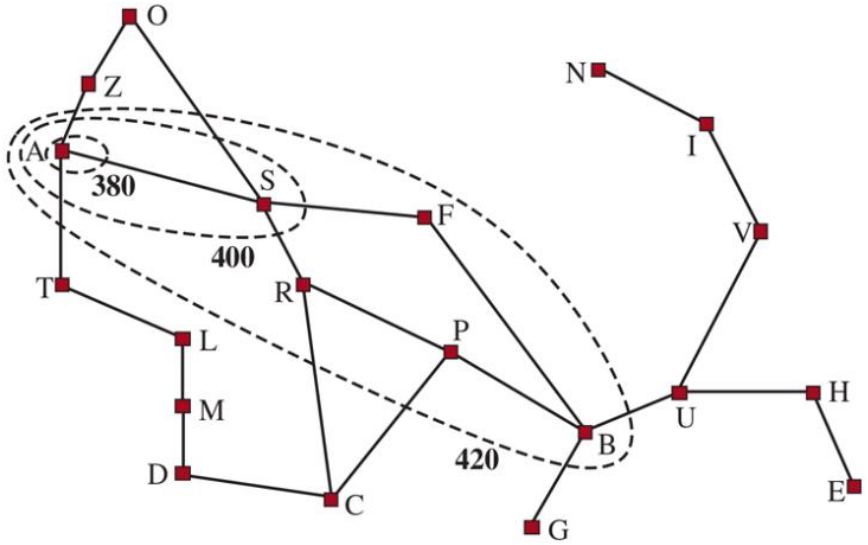
El algoritmo de búsqueda informada más común es la búsqueda A* search (A-star search), una búsqueda best-first que utiliza la función de evaluación:

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

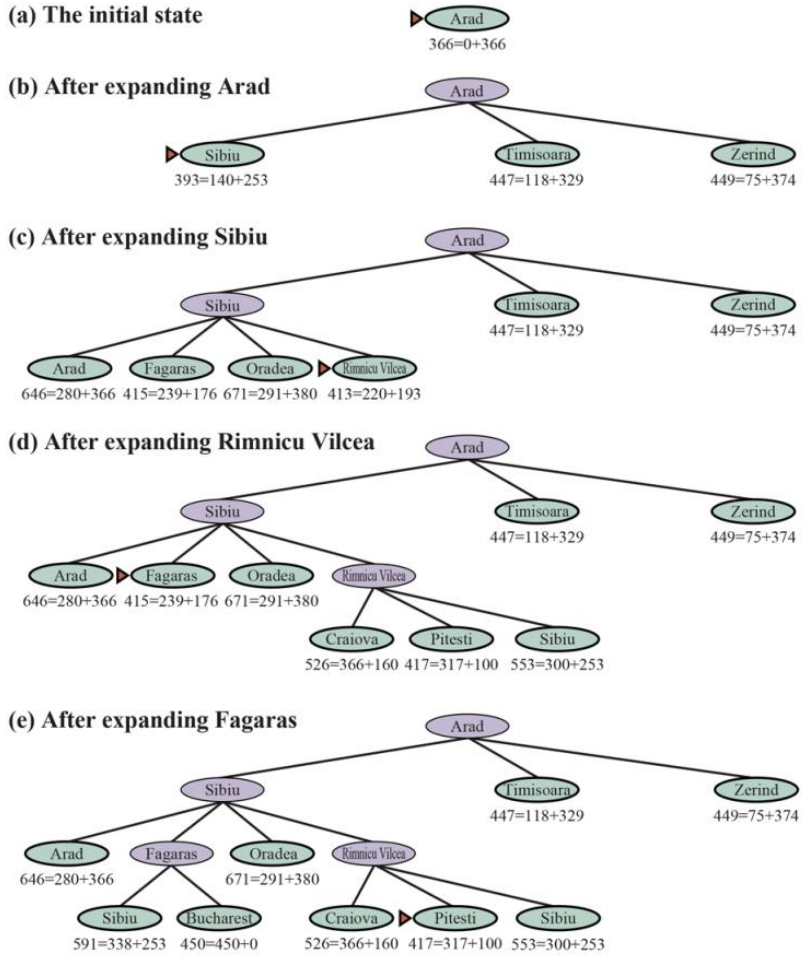
Donde *g(n)* es el costo del camino desde el estado inicial hasta el nodo *n*, y *h(n)* es el costo estimado del camino más corto desde n hasta un estado objetivo

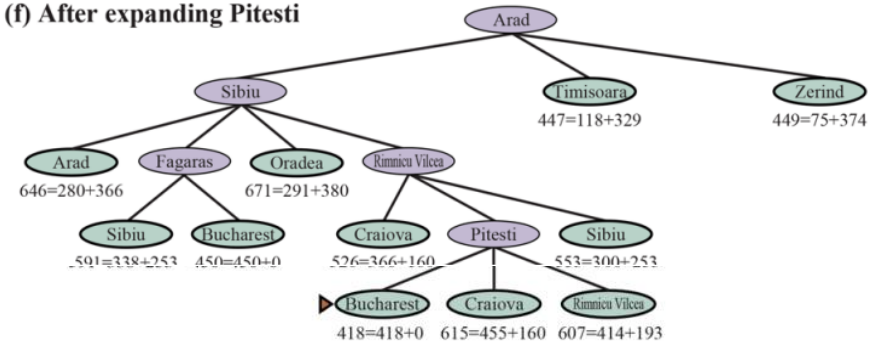
A* Search

Podemos simular el comportamiento del algoritmo de búsqueda dibujando contornos en el espacio de estados, de manera similar a los contornos en un mapa topográfico



A* Search


(a) The initial state
Arad
366=0+366
(b) After expanding Arad
Arad
Sibiu
Timisoara
Zerind
393=140+253
447=118+329
449=75+374
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
415=239+176
671=291+380
413=220+193
(d) After expanding Rimnicu Vilcea
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
415=239+176
671=291+380
Craiova
Pitesti
Sibiu
526=366+160
417=317+100
553=300+253
(e) After expanding Fagaras
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
671=291+380
Sibiu
Bucharest
Craiova
Pitesti
Sibiu
591=338+253
450=450+0
526=366+160
417=317+100
553=300+253


(f) After expanding Pitesti
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
671=291+380
Sibiu
Bucharest
Craiova
Pitesti
Sibiu
591=338+253
450=450+0
526=366+160
553=300+253
Bucharest
Craiova
Rimnicu Vilcea
418=418+0
615=455+160
607=414+193

A* Search

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

CARACTERISTICAS:

La búsqueda A* es **completa.**

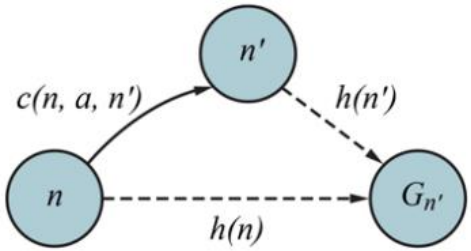
Que sea **optima** en costo depende de:

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:

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



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.

Optimalidad

“An algorithm is complete if it is guaranteed to find a solution if one exists, and it is optimal if it always finds the best solution.”

“Optimality is the property that the algorithm finds the best trajectory, provided one exists.”

“A* search is both complete and optimal, given an appropriate heuristic.”

Optimalidad

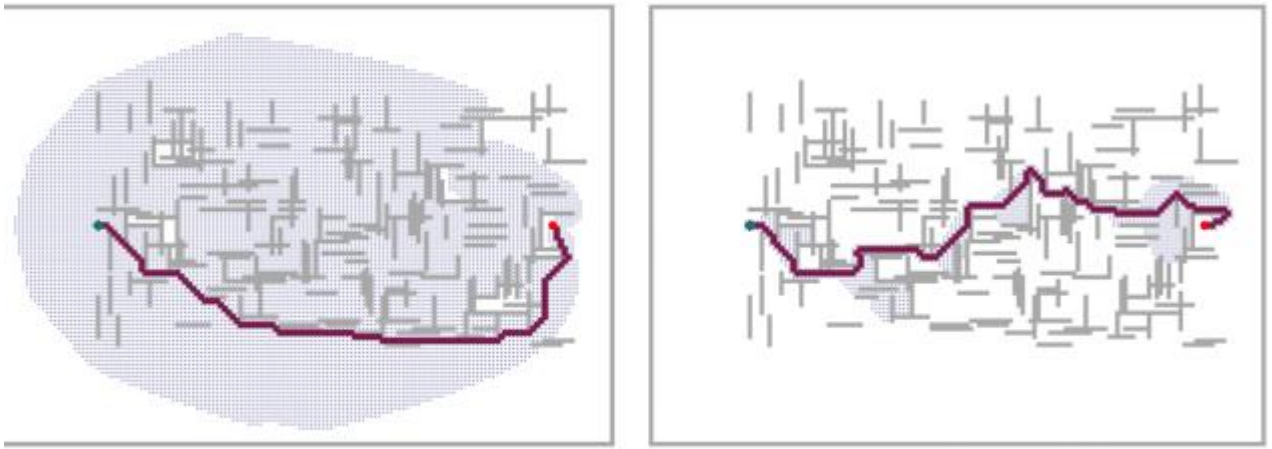
La búsqueda A* tiene muchas cualidades positivas, pero expande muchos nodos. Podemos explorar menos nodos (utilizando menos tiempo y espacio) si estamos dispuestos a aceptar soluciones que no sean óptimas, pero que sean "suficientemente buenas": lo que llamamos **soluciones satisfactorias.**

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

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

Optimalidad

En general, si la solución óptima cuesta *C**, una *Weighted A* Search* encontrará una solución que cuesta entre *C** y *WxC** ; pero en la práctica, usualmente obtenemos resultados mucho más cercanos a *C** que a *WxC**



Optimalidad

$$
\begin{aligned}
\text{A* search:} &\quad g(n) + h(n) &\quad (W = 1) \\
\text{Uniform-cost search:} &\quad g(n) &\quad (W = 0) \\
\text{Greedy best-first search:} &\quad h(n) &\quad (W = \infty) \\
\text{Weighted A* search:} &\quad g(n) + W \times h(n) &\quad (1 < W < \infty)
\end{aligned}
$$

Recursive best-first search

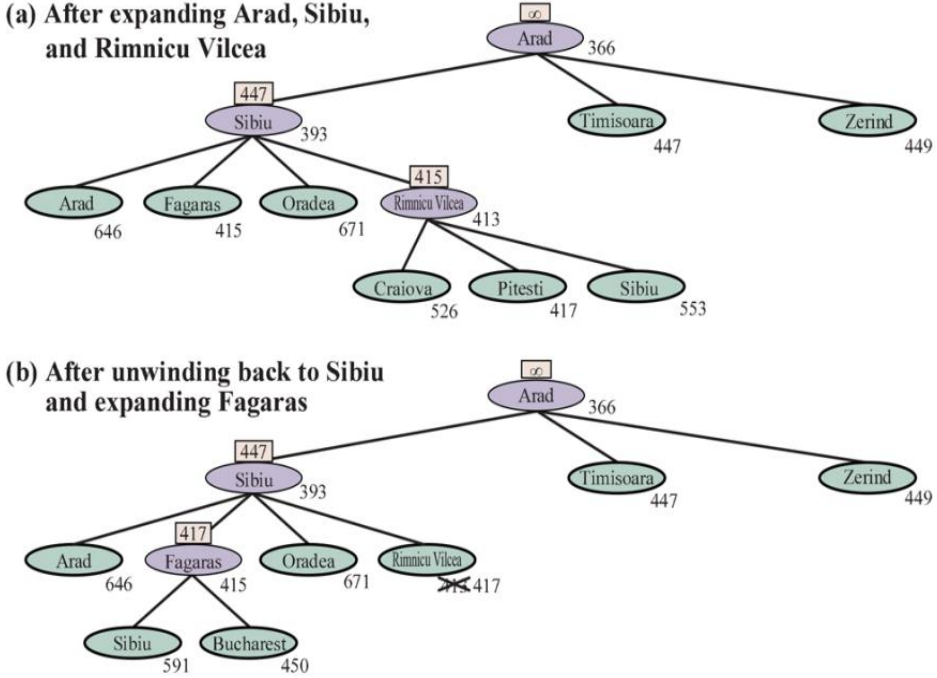
La memoria se divide entre la frontera y los estados alcanzados. En nuestra implementación de búsqueda mejor primero, un estado que está en la frontera se almacena en dos lugares: como un nodo en la frontera (para que podamos decidir qué expandir a continuación) y como una entrada en la tabla de estados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

Recursive best-first search

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

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



Recursive best-first search


```
function RECURSIVE-BEST-FIRST-SEARCH(problem) returns a solution or failure
    solution, fvalue ← 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 ← 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 ← max(s.PATH-COST + h(s), node.f))
    while true do
        best ← the node in successors with lowest f-value
        if best.f > f_limit then return failure, best.f
        alternative ← the second-lowest f-value among successors
        result, best.f ← RBFS(problem, best, min(f_limit, alternative))
        if result ≠ failure then return result, best.f
```

Recursive best-first search

UNIVERSIDAD EAFIT

Escuela de Ciencias Aplicadas e Ingeniería

CODE AND CLASS EXERCISE 1

Recursive best-first search

UNIVERSIDAD EAFIT

Escuela de Ciencias Aplicadas e Ingeniería

GENERAL CLASS EXERCISE 2

MAZE ALGORITHM

References

GeeksforGeeks. (2025, abril 7). *Informed Search Algorithms in Artificial Intelligence*. Recuperado de https://www.geeksforgeeks.org/artificial-intelligence/informed-search-algorithms-in-artificial-intelligence/

DeepAI. (s. f.). *Heuristics*. En *DeepAI machine learning glossary*. Recuperado de https://deepai.org/machine-learning-glossary-and-terms/heuristics

Berkeley University (CS188). (2022). *Completeness and Optimality – A Search**. En *CS188 Lecture Notes*. Recuperado de https://inst.eecs.berkeley.edu/~cs188/fa22/assets/notes/cs188-fa22-note02.pdf

Russell, S. J., & Norvig, P. (2020). Artificial intelligence: A modern approach (4th ed.). Pearson.