2. Búsqueda en espacio de estados

2.1 Introducción

La búsqueda en espacios de estados es una técnica fundamental en Inteligencia Artificial para resolver problemas. Consiste en explorar todas las posibles configuraciones (estados) que puede tener un sistema, aplicando reglas (operadores) que transforman un estado en otro, hasta encontrar uno que cumpla el objetivo (meta).

Ejemplo: En el ajedrez, los estados son las posiciones de las piezas en el tablero y las jugadas posibles son los operadores.

2.2 Tipos de métodos de búsqueda

Método	Descripción	Tipo de solución
Preciso	Específico para el problema, encuentra la solución óptima.	Exacta
Heurístico	Usa conocimientos del problema para aproximar la solución o aumentar eficiencia.	Aproximada
Metaheurístico	Generaliza procedimientos heurísticos para mejorar eficiencia, evitando quedarse en mínimos locales e intensificando la búsqueda en zonas prometedoras.	Aproximada

2.3 Conceptos Clave en Búsqueda de Soluciones

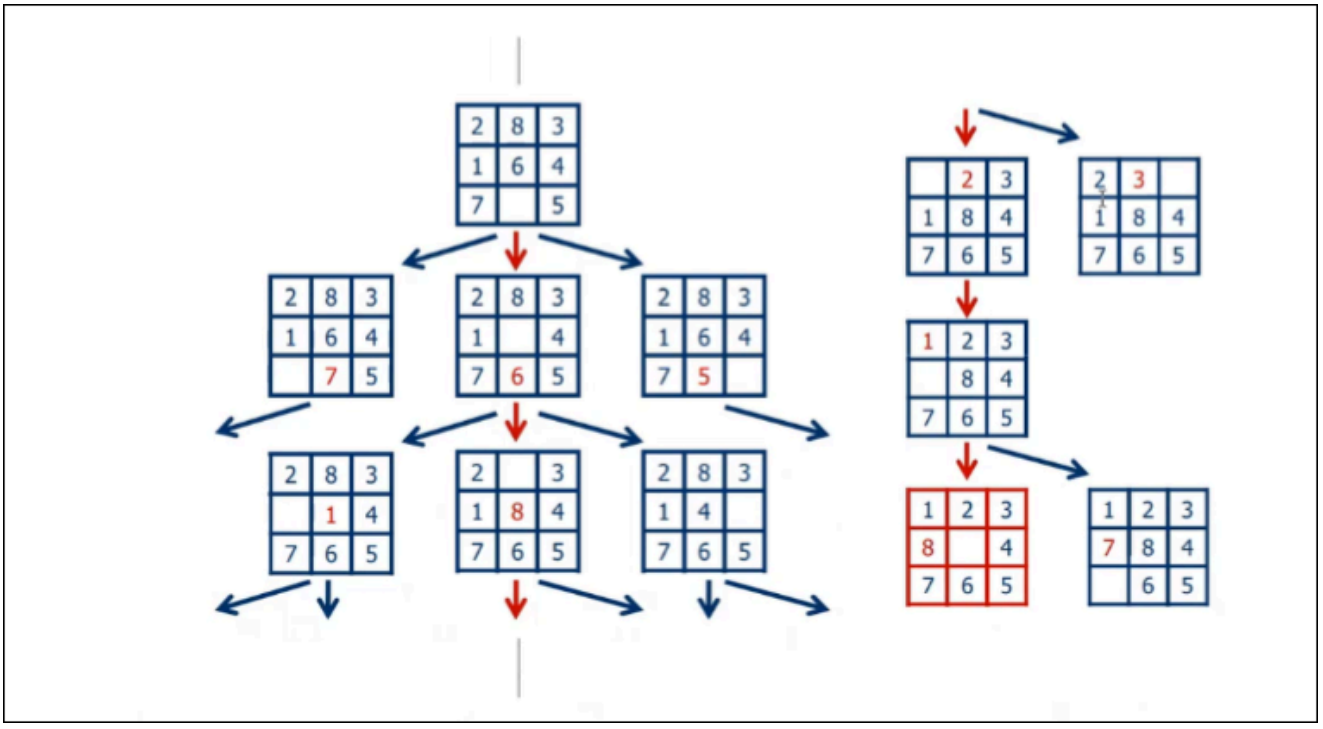
La resolución de problemas mediante búsqueda implica modelar el problema en términos formales. Los conceptos fundamentales son:

- **Estado:** Representa una configuración posible del sistema (por ejemplo, cómo están las piezas en un tablero).
- **Operador:** Acción que transforma un estado en otro (como mover una pieza, cambiar una letra, etc).
- **Secuencia de estados:** Camino que enlaza el estado inicial con el estado meta a través de operadores.
- **Prueba de meta:** Permite verificar si un estado cumple el objetivo deseado.
- **Condición de parada:** Criterio que determina cuándo se debe detener la búsqueda (por ejemplo, alcanzar la meta o agotar las posibilidades).

2.4 Ejemplos de Modelado de Problemas

Ejemplo 1: 8-puzzle (8-quebracabezas)

- **Estados:** Posiciones de las piezas en el tablero.
- **Operadores:** Movimiento del espacio vacío en dirección ↑, ↓, →, ←.
- **Meta:** Lograr la configuración final (orden correcto de piezas).
- **Criterio de coste:** Número de movimientos realizados.



Ejemplo 2: Criptoaritmética

- **Estados:** Asignación de dígitos a letras.
- **Operadores:** Cambiar una letra por un dígito (sin repetir dígitos).
- **Meta:** Todas las letras convertidas y la operación aritmética resulta correcta.
- **Criterio de coste:** Generalmente cero, solo interesa la validez de la solución.

2.5 Espacio de Estados y Grafos de Búsqueda

- El **espacio de estados** es el conjunto de todas las configuraciones alcanzables desde el estado inicial, aplicando cualquier secuencia de operadores.
- La **búsqueda** consiste en encontrar un estado meta y, opcionalmente, la ruta para alcanzarlo.
- Muchas veces, este espacio se representa mediante un **grafo**, donde los **nodos** son los estados y las **aristas** las acciones.

Consideraciones en Problemas Complejos

- En problemas grandes, es inviable explorar todas las alternativas.
- Cuando hay varias soluciones, basta con encontrar una que sea "suficiente".

- Se emplean **heurísticas** para guiar la búsqueda hacia los caminos más prometedores.

> **Heurística:** Es un criterio basado en conocimiento del problema que ayuda a seleccionar los operadores o acciones más prometedoras, descartando opciones poco útiles.
> **Una heurística es como un "atajo inteligente" que evita búsquedas exhaustivas.**

2.6 Estrategias de búsqueda a ciegas

Las **estrategias de búsqueda a ciegas** (no informadas) exploran posibles soluciones sin información adicional sobre cuál camino puede ser mejor. Se utilizan principalmente en problemas donde no se tiene una heurística clara para guiar la búsqueda.

2.6.1 Propiedades y Terminología

- **Búsqueda completa:** garantiza encontrar una solución si existe.
- **Búsqueda óptima:** garantiza encontrar la mejor solución disponible.
- **Complejidad temporal:** número total de nodos explorados durante la búsqueda.
- **Complejidad espacial:** número máximo de nodos almacenados en memoria simultáneamente.

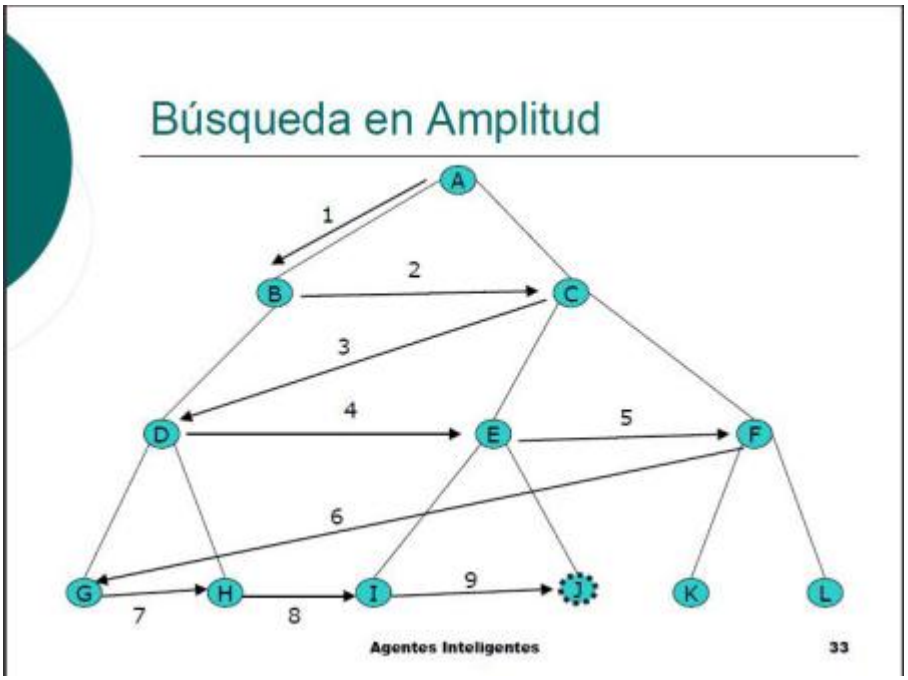
Símbolos clave:

- **r**: factor de ramificación (promedio de sucesores por nodo)
- **p**: profundidad de la solución
- **m**: profundidad máxima
- **l**: límite de profundidad

2.6.2 Tipos de Estrategias

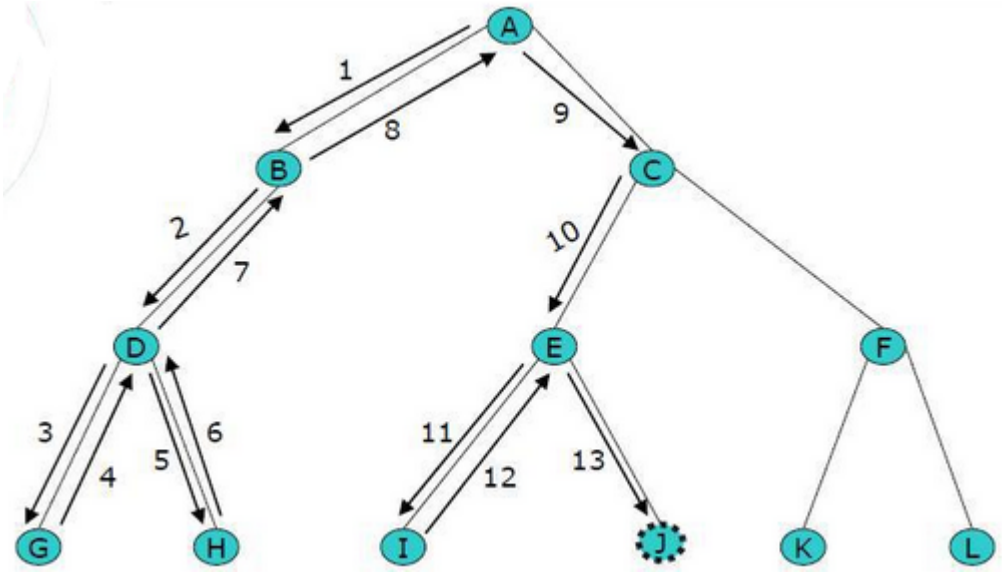
1. Búsqueda en Amplitud (Breadth-First Search)

- Explora primero los nodos más cercanos al inicial (por niveles).
- **Ventajas:** Completa y óptima.
- **Desventajas:** Alta complejidad espacial y temporal.

- 

2. Búsqueda en Profundidad (Depth-First Search)

- Explora lo más profundo posible antes de retroceder (backtracking).
- **Ventajas:** Uso eficiente de memoria.
- **Desventajas:** No siempre es completa ni óptima.

- 

3. Búsqueda en Profundidad Limitada

- Se establece un límite máximo de profundidad para la exploración.
- Evita ciclos infinitos, pero puede perder soluciones si el límite es bajo.

4. Búsqueda en Profundidad Iterativa

- Aplica búsqueda en profundidad con límites crecientes, combinando ventajas de amplitud y profundidad.
- Es completa y óptima (en espacios finitos).

5. Búsqueda Bidireccional

- Realiza la búsqueda simultáneamente desde el estado inicial y desde el objetivo, esperando que ambas se encuentren.
- Reduce la complejidad temporal y espacial.

2.6.3 Comparativa de Estrategias

Estrategia	Completa	Óptima	Tiempo	Espacio
Amplitud	Sí	Sí	O(r^p)	O(r^p)
Profundidad	No*	No	O(r^m)	O(r*m)
Prof. limitada	No*	No	O(r*l)	O(r*l)
Iterativa	Sí	Sí	O(r^p)	O(r^p)
Bidireccional	Sí	Sí	O(r^(p/2))	O(r^(p/2))

* Solo completa en espacios finitos y sin bucles.

2.7 Búsqueda Heurística

La **búsqueda heurística** es una técnica fundamental de la inteligencia artificial para encontrar soluciones "buenas" en problemas complejos, donde explorar todas las posibilidades sería inviable. A diferencia de las estrategias ciegas, utiliza información relevante del problema para guiar la exploración y suele ser mucho más eficiente.

2.7.1 ¿Cómo funciona la búsqueda heurística?

La búsqueda heurística evalúa cada posible estado (o nodo) utilizando una **función de evaluación**, que combina:

- $g(n)$: El coste real acumulado desde el estado inicial hasta el nodo actual.
- $h(n)$: Una estimación heurística del coste que falta para llegar desde el nodo actual a la solución (generalmente es una distancia).

La función de evaluación se expresa como:

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

Esto nos permite priorizar los nodos que parecen más prometedores, equilibrando el progreso real y la estimación futura.

2.7.2 Tipos de búsqueda según la función de evaluación

- **Búsqueda de coste uniforme:**
  Solo se tiene en cuenta el coste real recorrido.

$$f(n) = g(n)$$

- **Búsqueda avara ("greedy"):**
  Solo se considera la estimación heurística, buscando el nodo que parece más cerca de la meta, sin considerar el coste recorrido.

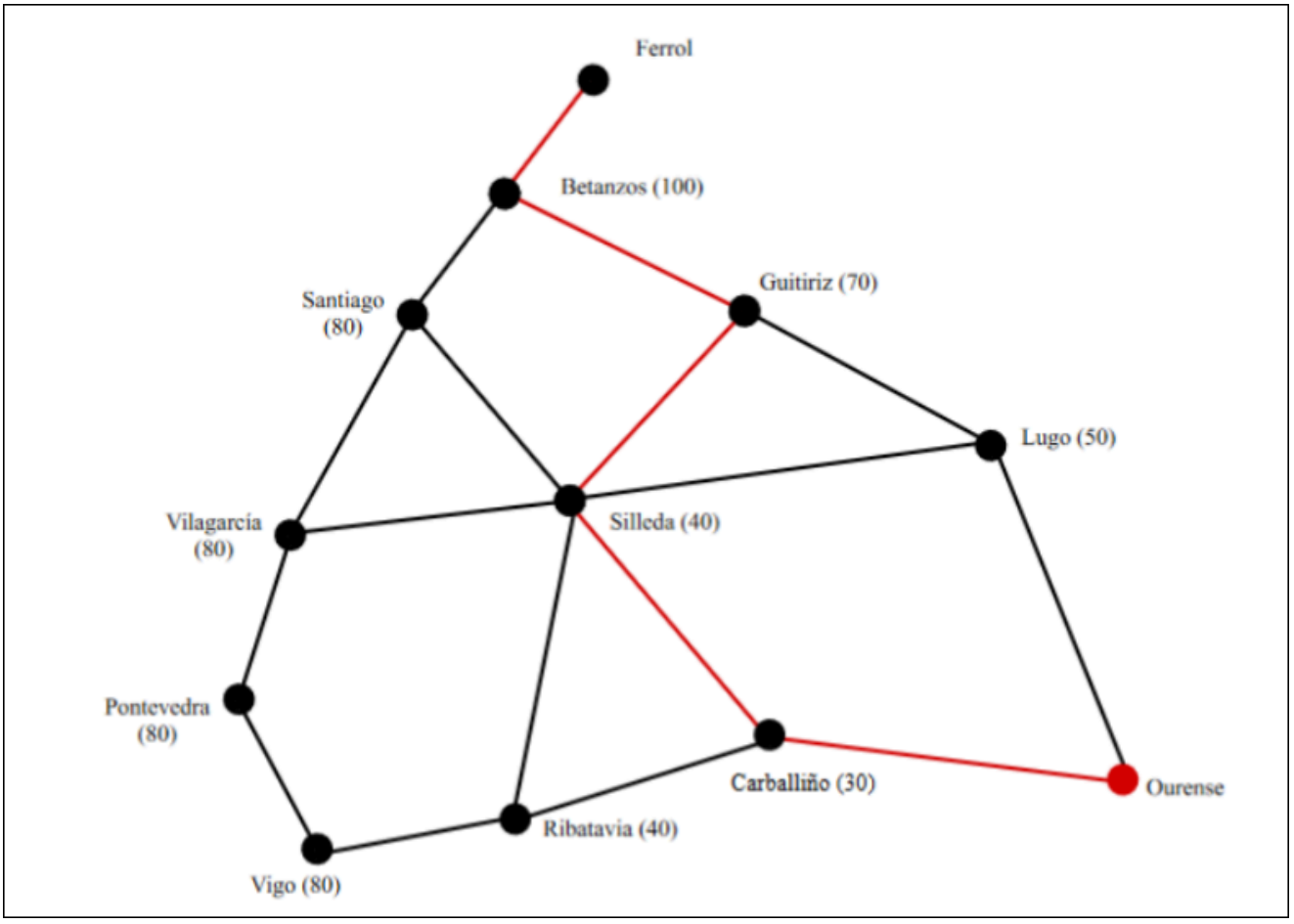
$$f(n) = h(n)$$

- **Algoritmo A*:**
  Combina ambos factores, buscando el camino más prometedor y eficiente.

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

2.7.3 Ejemplo práctico: Ruta entre ciudades

Problema: Ir de Ferrol a Ourense minimizando la distancia total.



Datos del problema:

- $g(n)$: Distancia real recorrida desde Ferrol hasta la ciudad actual.
- $h(n)$: Distancia en línea recta desde la ciudad actual hasta Ourense (heurística optimista).

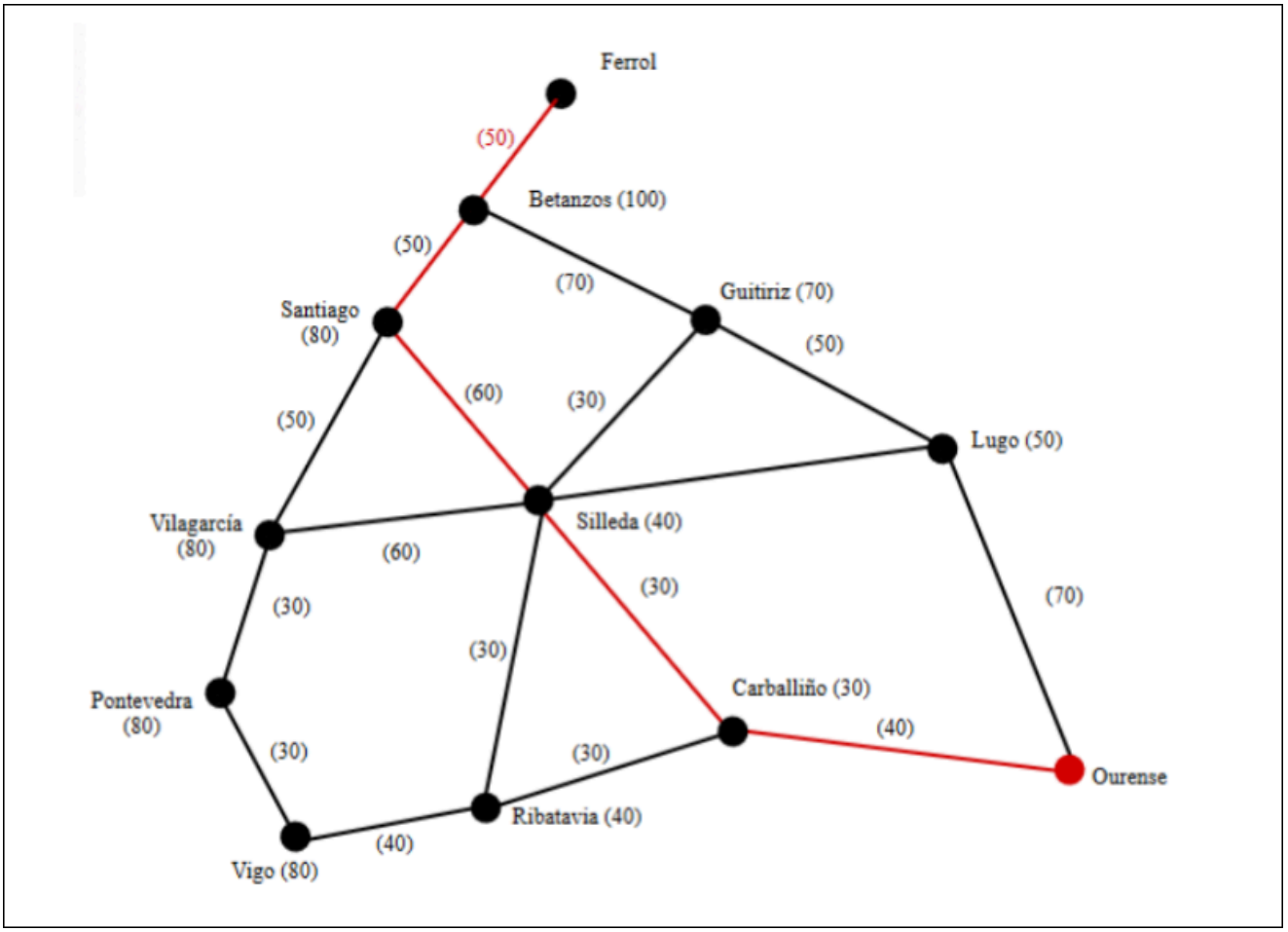
Ejemplo de cálculo con el algoritmo A*:

Supón que desde Betanzos puedes ir a Santiago o a Guitiriz.
Calculamos para cada opción:

$$f(Guitiriz) = g(Guitiriz) + h(Guitiriz) = 120 + 70 = 190$$

$$f(Santiago) = g(Santiago) + h(Santiago) = 100 + 80 = 180$$

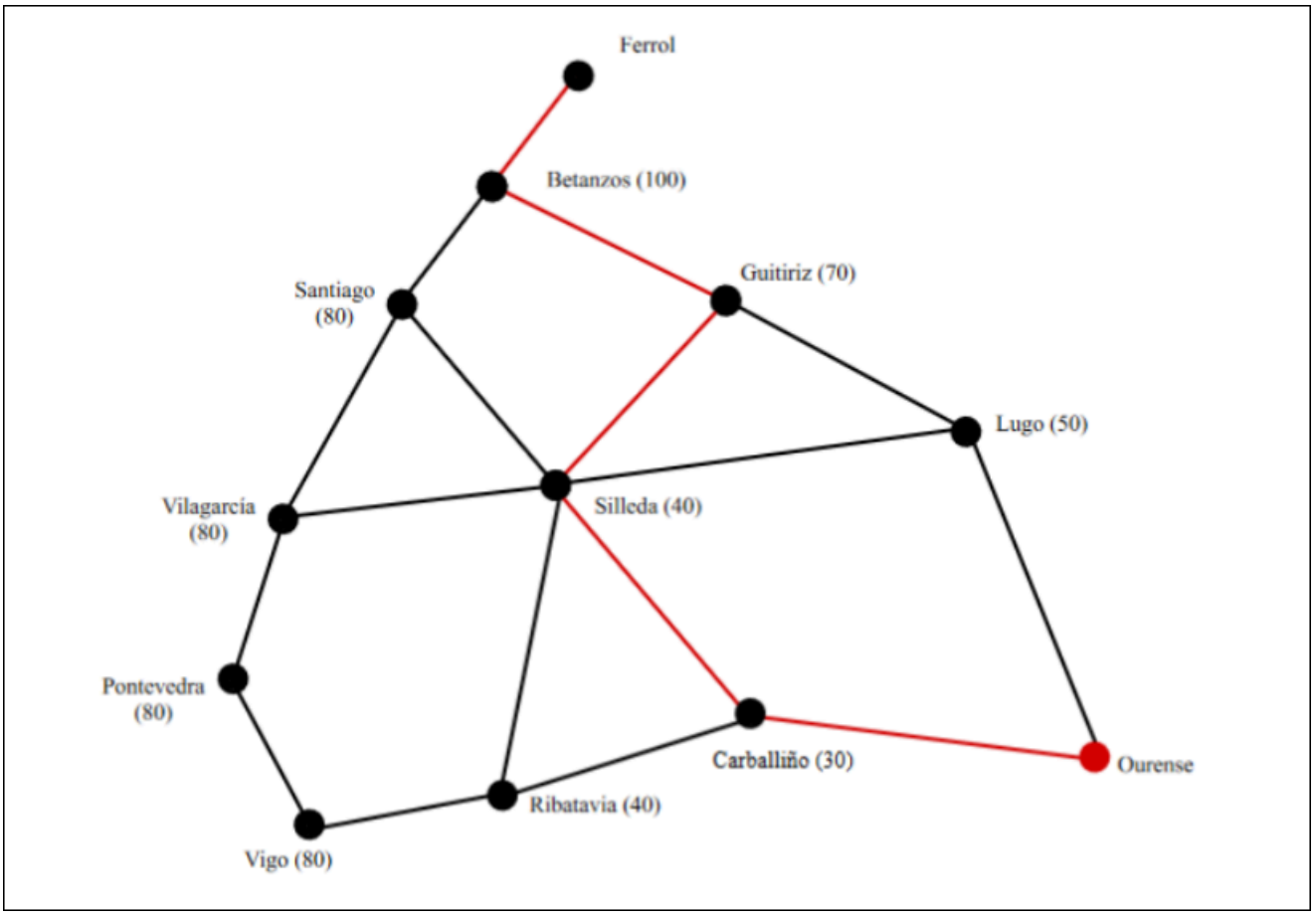
Elegimos Santiago porque tiene menor f(n). Repitiendo este proceso en cada paso, encontramos la ruta más eficiente.



Resultado: El algoritmo A* encuentra una distancia total de 230 km.

Búsqueda avara

Si solo usamos la heurística (distancia en línea recta), podríamos elegir caminos que parecen más cortos, pero pueden resultar menos eficientes en la realidad.



Resultado: La distancia total recorrida es mayor, 240 km.

Conclusión:
En general, A* suele ofrecer mejores resultados que la búsqueda avara, aunque depende de la calidad de la heurística utilizada.

2.7.4 ¿Qué es una heurística?

Una **heurística** es una función que estima el coste que falta para alcanzar la solución desde un estado dado.
Su objetivo es orientar la búsqueda hacia los caminos más prometedores y descartar opciones menos útiles.
Las heurísticas **admisibles** (optimistas) nunca sobrestiman el coste real que falta, lo que puede garantizar soluciones óptimas cuando se usa A*.

2.7.5 Ejemplos de heurísticas en el 8-puzzle

Para resolver el 8-quebracabezas, podemos usar distintas heurísticas:

1. **Número de piezas fuera de lugar:**
   Cuenta cuántas piezas están en posición incorrecta.
   Es una estimación sencilla pero útil.

2. **Distancia Manhattan:**
   Suma la distancia (en movimientos permitidos) entre cada pieza y su posición

objetivo, ignorando las demás piezas.

Si tienes varias heurísticas admisibles, puedes combinar ambas tomando el valor mayor para cada estado, obteniendo una estimación más precisa y aún admisible.