Trabajo Práctico N° 1

"Algoritmos de búsqueda en la Torre de Hanoi"

Facultad de Ingeniería de la Universidad de Buenos Aires
Laboratorio de Sistemas Embebidos
Introducción a Inteligencia Artificial
David Canal
Marzo 2023

1. Consigna de Trabajos

En clase presentamos el problema de la torre de Hanoi. Además, vimos diferentes algoritmos de búsqueda que nos permitieron resolver este problema. Para este trabajo práctico, deberán implementar un método de búsqueda para resolver con 5 discos, del estado inicial y objetivo que se observa en la siguiente imagen:

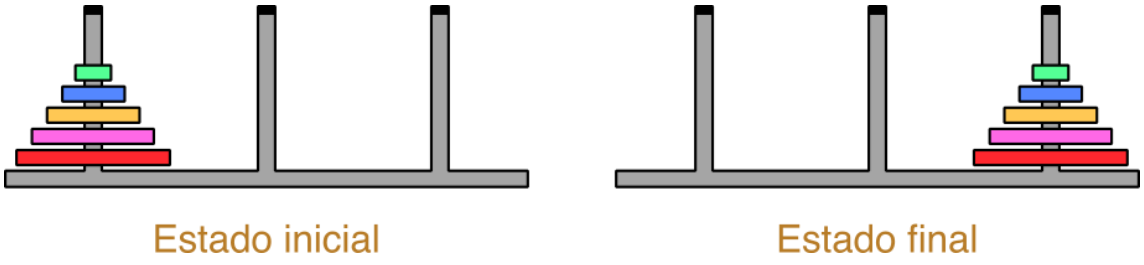

Figura 1.1:" Estado inicial y final de la Torre de Hanoi".

(1)¿Cuáles son los PEAS de este problema?

(2) ¿Cuáles son las propiedades del entorno de trabajo?

(3) En el contexto de este problema, establezca cuáles son los: estado, espacio de estados, árbol de búsqueda, nodo de búsqueda, objetivo, acción y frontera.

(4) Implemente algún método de búsqueda.

(5) ¿Qué complejidad en tiempo y memoria tiene el algoritmo elegido?

(6) A nivel implementación, ¿qué tiempo y memoria ocupa el algoritmo? (Se recomienda correr 10 veces y calcular promedio y desvío estándar de las métricas).

(7) Si la solución óptima es 2^k-1 movimientos con k igual al número de discos. Qué tan lejos está la solución del algoritmo implementado de esta solución óptima (se recomienda correr al menos 10 veces y usar el promedio de trayecto usado).

2. Resolución

(1) PEAS del problema de la Torre de Hanoi

- **Performance:** El rendimiento se mide por la cantidad de movimiento necesarios para resolver el problema. Para la Torre de Hanoi, el objetico es mover todos los discos de la torre de origen a la torre destino, utilizando la torre intermedia como auxiliar.

- **Eviroment:** El entorno en este caso es la representación de las tres torres y los discos sobre ellas. Cada disco es de un tamaño diferente y puede ser movido de una torre a otra, siguiendo la regla de que un disco más grande nunca puede estar encima de uno más pequeño.

- **Actuators:** En nuestro caso, los actuadores son las acciones del movimiento de los discos de una torre a otra. Estos movimientos pueden ser de tres tipos, mover un disco de la torre A a la torre B, de la torre a la torre C, o de la torre B a la torre c (y viceversa).

- **Sensors:** En este caso el agente necesita saber el estado actual de las torres, es decir, cuántos discos hay en cada una y qué tamaño tiene. Esto permite determinar cuál es el siguiente movimiento válido.

(2) Propiedades del entorno de trabajo

El entorno de trabajo es:

- **Totalmente observable.** Porque el agente tiene acceso al estado completo de las torres en todo momento (conoce las posiciones de los discos en cada torre).
- **Determinista.** Porque el resultado de mover un disco a una torre está completamente determinado por la acción del agente y el estado actual de las torres.
- **Secuencial.** Dado que, si se toma una serie de movimientos para resolver la torre de hanoi, cada movimiento depende del anterior para lograr la solución.
- **Estático.** En la torre de hanoi, una vez que se decide un movimiento, las torres no cambian hasta que se realice la siguiente acción.
- **Discreto.** En este caso, tanto el estado del juego (posición de los discos en las torres) como las acciones del agente (mover un disco de una torre a otra) son discretas y tienen valores finitos.
- **Individual.** Porque estamos considerando un entorno de agente individual, ya que existen un número infinito de posibles posiciones de los discos o movimientos posibles.

(3) Estado, espacio de estados, árbol de búsqueda, nodo de búsqueda, objetivo, acción y frontera.

- **Estado:** Es la configuración actual de los discos en las tres varillas.
- **Espacio de estados:** Son todas las posibles combinaciones de discos en las tres varillas, respetando la restricción de que un disco más grande nunca puede estar sobre un más pequeño.
- **Árbol de búsqueda:** Cada nodo en el árbol representa un estado, y las ramas representan las acciones que llevan de un estado a otro.
- **Nodo de búsqueda**: Es una configuración particular de los discos en las varillas.
- **Objetivo**: es mover todos los discos de la varilla de origen a la varilla objetivo, respetando las reglas del juego.

(4) Implementación de un método de Búsqueda

Se implementó un algoritmo de búsqueda no informada como "Búsqueda Primero en Profundidad", el cual fue presentado en clase durante el estudio del tema. Este algoritmo avanza expandiendo el nodo más profundo en la frontera actual del árbol de búsqueda, removiendo los nodos de la frontera a medida que son explorados.

En la Figura 2.1 se presenta la función utilizada en la implementación del algoritmo de búsqueda en profundidad en python.

```
# Agregado de David Canal
##########################
# Algorítmo de búsqueda primero en profundidad
def best_frist_search(Problem, display: bool = False):

    node_root = NodeHanoi(Problem.initial)


    frontier = [node_root]
    explored = []

    while len(frontier) > 0:
        node = frontier.pop()  # Extraemos el primer nodo de la cola
        explored.append(node.state)
        print(node)
        if Problem.goal_test(node.state):
            last_node = node
            print("Encontramos la solución")
            return node

        for next_node in node.expand(Problem):
            if next_node.state not in explored:
                frontier.append(next_node)
    return None
```

Figura 2.1: "Función best_frist_search".

(5) Complejidad en tiempo y memoria del algoritmo implementado

- **El tiempo de ejecución:** 0,1448s aproximadamente.
- **Máxima memoria requerida:** 0.37 MB.

Punto (6) y (7)

Se realizaron 10 simulaciones para realizar las siguientes aseveraciones:

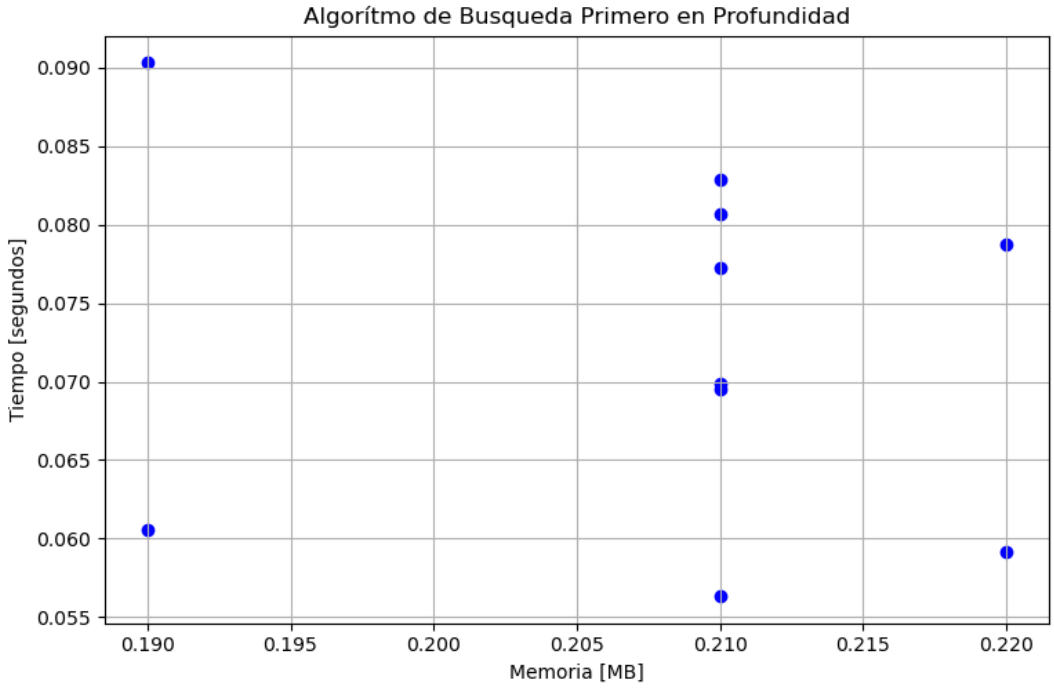

Figura 2.2: "Tiempo de ejecución vs Memoria requerida".

Variable	Media	Desviación estándar
Tiempo [seg]	**0,07253**	**0,01133**
Memoria [MB]	**0,2080**	**0,01033**

Tabla 2.1: "Estadísticas de variables de estudio".

$$Tiempo = (0{,}07253 \pm 0{,}003583)\ seg$$

$$Memoria = (0{,}2080 \pm 0.003267)\ MB$$

Además, se observó que la trayectoria obtenida en las 10 simulaciones constó de 121 movimientos. En comparación con la solución óptima, que requiere de $2^5-1 = 31$ movimientos para resolver el problema de la Torre de Hanoi con 5 disco, se encontró una diferencia de 90 movimientos adicionales en promedio. Esto implica que la estrategia utilizada en las simulaciones requirió un esfuerzo mayor para alcanzar la solución en comparación con la solución teóricamente óptima.