Inteligencia Artificial 2026

Lab 06

13.abril.2026

En este laboratorio vamos a implementar la solución de Problemas de Satisfacción de Restricciones (CSP), mediante búsqueda DFS con *backtracking* y con *forward checking*.

1. **Sudoku**:
   Implementar y comparar dos estrategias de búsqueda en profundidad (DFS) para resolver un Sudoku de $n \times n$ (n debe ser un cuadrado ≥ 4):

   - Backtracking: retrocede cuando encuentra una incompatibilidad de restricciones.
   - Forward Checking (Filtering): anticipa fallos eliminando valores del dominio de variables no asignadas.

   Estructura del Árbol de Búsqueda: En este problema, el árbol se define de la siguiente manera:

   - Estado Raíz: El tablero inicial con las posiciones fijas.
   - Nodos: Tableros parcialmente llenos.
   - Hijos: El resultado de asignar un número válido (1 a 9) a la siguiente celda vacía.
   - Hojas: Un tablero completo (solución) o un tablero donde no hay valores legales para una celda (poda).

   Especificaciones: Deberán implementar la clase `SudokuSolver`. El usuario definirá el tamaño n (donde $\sqrt{n}$ debe ser entero para las subregiones) y el tablero inicial. **Requerimientod de la clase:**

   - `__init__(self, n, board)`: Inicializa el tamaño y el estado del tablero.
   - `is_valid(self, row, col, num)`: Verifica si un número puede ir en esa posición.
   - `solve_backtracking()`: Implementación de DFS pura con backtracking.
   - `solve_filtering()`: Implementación de DFS con Forward Checking.
   - `display()`: Imprime el tablero de forma legible.

2. Resolver un tablero de Sudoku de su elección, mediante *backtracking* y mediante *forward checking*, para los siguientes casos:

   - un sudoku de 4×4
   - un sudoku de 9×9 nivel fácil
   - un sudoku de 9×9 nivel extremo
   - un sudoku de 16×16.

   Para cada caso, mostrar la solución encontrada por cada método, así como el número de nodos visitados y el tiempo de ejecución de cada algoritmo.
   Compare a partir de estos escenarios cuál es más eficiente.

3. **N Queens:**
   Colocar N reinas en un tablero de ajedrez de $N \times N$ de tal manera que ninguna reina amenace a otra. Para ello, usaremos una representación de tipo permutaciones, esto es, un arreglo unidimensional a de tamaño N: El índice del arreglo representa la fila, y el valor `a[i]` representa la columna de esa fila donde se coloca la pieza

   De nuevo, deberá implementar y comparar dos estrategias de búsqueda en profundidad (DFS) para resolver el problema de N reinas:

   Especificaciones: Deberán implementar la clase `NQueensSolver`, el cual devuelve una solución para el problema de N Reinas. El usuario definirá el tamaño N y el tablero inicial ser asume vacío.

   Requerimientod de la clase:

- `__init__(self, n)`: Inicializa el tamaño y el estado del tablero.
- `is_valid(self, row, col)`: Verifica si hay ataques de otras piezas en el tablero a esta casilla.
- `solve_backtracking()`: Implementación de DFS pura con backtracking.
- `solve_filtering()`: Implementación de DFS con Forward Checking.
- `display()`: Imprime el tablero de forma legible.

4. Resolver un problema de N reinas, mediante backtracking y mediante forward checking, para los siguientes casos:

   - $N = 4$,
   - $N = 8$,
   - $N = 12$,
   - $N = 15$.

   Para cada caso, mostrar la solución encontrada por cada método, así como el número de nodos visitados y el tiempo de ejecución de cada algoritmo.
   Compare a partir de estos escenarios cuál es más eficiente.

5. Modificar alguno de los algoritmos para obtener TODAS las soluciones de un problema de N reinas, y usarlo para resolver los casos:

   - $N = 4$,
   - $N = 5$,
   - $N = 6$.

   ¿Cuántas soluciones posibles hay en cada caso?

Pregunta de discusión:

¿Cuál enfoque es más eficiente, *backtracking* o *forward checking*? Explique con sus palabras por qué considera que esta estrategia es más eficiente que la otra. (Se espera una explicación técnica computacional que justifique su discusión).

Recursos para el Sudoku

Pueden usar `https://sudokubliss.com/` para generar ejemplos de sudokus, de diferentes niveles. En la parte inferior aparece un botón `solver` para resolver el sudoku de forma automática (el cual probablemente use algunos de los algoritmos que están implementando).