**Segunda Entrega Proyecto**


**Santiago Hernández Barbosa**

**Juan Diego Pardo**

**26 de septiembre 2025**

**Introducción a Inteligencia Artificial**

**Ing. Julio Omar Palacio Niño**

## Introducción

El presente proyecto aborda la resolución del Sudoku desde la perspectiva de la satisfacción de restricciones (CSP). En este enfoque, cada celda del tablero se trata como una variable con un dominio de valores posibles y se aplican restricciones simples: no repetir números en filas, columnas ni subcuadrículas. Este marco permite organizar la búsqueda de soluciones de manera ordenada, comparando distintos métodos de exploración y poda del espacio de búsqueda.

El objetivo central es implementar y evaluar tres variantes de backtracking: (i) fuerza bruta (sin heurísticas), (ii) backtracking básico y (iii) backtracking con comprobación hacia adelante (forward checking). El alcance se limita a resolver tableros de 9×9 proporcionados como entrada y a medir el desempeño de cada método mediante dos métricas claras: tiempo de ejecución e intentos de asignación.

El proyecto ofrece un código sencillo y legible, estructurado de forma procedural y con nomenclatura coherente, junto con una comparación empírica que muestra ventajas y límites de cada método. Además, se entregan lineamientos prácticos para escoger la variante más adecuada según las características del tablero.

## Planteamiento del problema (CSP)

El Sudoku puede entenderse como un problema de satisfacción de restricciones (CSP, por sus siglas en inglés). En este tipo de problemas se definen tres componentes fundamentales: variables, dominios y restricciones.

En el caso del Sudoku:

- **Variables:** cada una de las 81 celdas del tablero de 9×9 se considera una variable independiente.
- **Dominios:** a cada variable se le asigna como dominio el conjunto de números del 1 al 9. Cuando una celda ya tiene un valor dado por el enunciado, su dominio se restringe a ese único número.
- **Restricciones:** las variables deben cumplir tres condiciones básicas:
    1. No repetir valores en una misma fila.
    2. No repetir valores en una misma columna.
    3. No repetir valores en una misma subcuadrícula de 3×3.

Estas condiciones hacen que el Sudoku sea un ejemplo natural de CSP, ya que el objetivo consiste en asignar un valor válido a cada variable de forma que todas las restricciones se satisfagan simultáneamente. Esta formulación permite aplicar métodos de búsqueda sistemática y comparar su eficiencia.

En el código implementado, el tablero se representa como una matriz de 9×9, donde el valor **0** indica una celda vacía y los números del 1 al 9 representan valores asignados. Esta notación facilita tanto la lectura como la manipulación del tablero durante el proceso de resolución.

**Métodos implementados**

En el desarrollo del proyecto se implementaron tres enfoques distintos para resolver el Sudoku, todos basados en el uso de backtracking, pero con variaciones que permiten observar cómo pequeñas mejoras reducen el espacio de búsqueda y el tiempo de cómputo.

**1. Fuerza bruta (backtracking sin heurísticas).**
Este método consiste en recorrer el tablero celda por celda y probar, de manera secuencial, todos los números posibles entre 1 y 9. Cada vez que una asignación resulta inválida según las reglas del Sudoku, el algoritmo retrocede y prueba un valor diferente. Aunque garantiza encontrar una solución, su complejidad es muy alta, ya que en el peor de los casos explora un número exponencial de combinaciones. Por ello, se utiliza principalmente como línea base de comparación.

**2. Backtracking básico.**
A diferencia de la fuerza bruta, en esta variante cada asignación se valida inmediatamente antes de continuar. Es decir, solo se avanza si el número colocado en una celda cumple las restricciones de fila, columna y subcuadrícula. Esta verificación temprana evita expandir ramas que ya son incorrectas, reduciendo el número de intentos necesarios y mejorando la eficiencia del método.

**3. Backtracking con Forward Checking (BT+FC).**
En esta versión se añade un mecanismo de propagación de restricciones. Cada vez que se asigna un valor a una celda, dicho valor se elimina de los dominios posibles de todas las celdas relacionadas (misma fila, columna y subcuadrícula). Si alguna celda queda sin valores posibles, el algoritmo detecta inconsistencia de inmediato y realiza backtrack. Esta estrategia, combinada con la selección de la celda con menos valores disponibles (heurística MRV), permite reducir significativamente el espacio de búsqueda y agilizar la resolución en tableros con un alto nivel de dificultad.

Estos tres métodos muestran cómo la incorporación gradual de técnicas de poda y heurísticas transforma un enfoque muy costoso en uno más eficiente, sin perder la claridad de la solución paso a paso.

**Diseño**

- solucion1_FB — No se utiliza del todo fuerza bruta, se combina con un poco de heuristica para calcular la solución exacta en cada espacio para resolver el sudoku
- solucion2_BT — Se utiliza una búsqueda con retroceso inteligente

- solucion3_BT_FC (backtracking con MRV + Forward Checking) — Mejora la eficiencia reduciendo el espacio de búsqueda eliminando valores inválidos antes de hacer las elecciones

**Casos de Prueba**

**Tablero ya resuelto**

- Entrada: Sudoku completo y válido.
- Resultado esperado: el algoritmo detecta que no hay espacios vacíos y retorna el tablero sin modificaciones, con muy pocos intentos.

**Tablero con una sola celda vacía**

- Entrada: Sudoku con 80 casillas llenas y solo 1 en 0.
- Resultado esperado: el algoritmo encuentra la respuesta de inmediato.

**Tablero inválido**

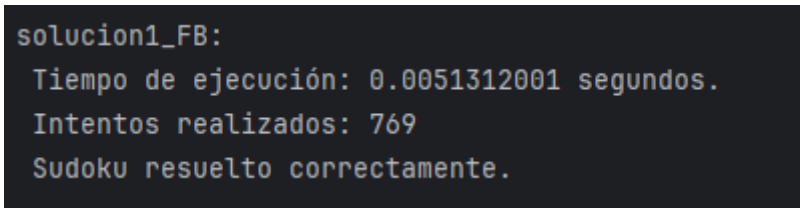
- Entrada: Sudoku con dos números repetidos en la misma fila.
- Resultado esperado: el algoritmo debe retornar que no existe solución.

**Tablero con múltiples soluciones**

- Entrada: Sudoku mal planteado que permite más de una solución.
- Resultado esperado: el algoritmo retorne al menos una solución válida o indique que hay más de una, según la implementación.

**Resultados**

**Solución 1**




**Solución 2**

```
solucion2_BT:
 Tiempo de ejecución: 0.0056688000 segundos.
 Intentos realizados: 78
 Sudoku resuelto correctamente.
```

**Solución 3**

```
solucion3_BT_FC:
 Tiempo de ejecución: 0.0011372000 segundos.
 Intentos realizados: 0
 Sudoku resuelto correctamente.
```

**Solución**

```
[3, 1, 6, 5, 7, 8, 4, 9, 2]
[5, 2, 9, 1, 3, 4, 7, 6, 8]
[4, 8, 7, 6, 2, 9, 5, 3, 1]
[2, 6, 3, 4, 1, 5, 9, 8, 7]
[9, 7, 4, 8, 6, 3, 1, 2, 5]
[8, 5, 1, 7, 9, 2, 6, 4, 3]
[1, 3, 8, 9, 4, 7, 2, 5, 6]
[6, 9, 2, 3, 5, 1, 8, 7, 4]
[7, 4, 5, 2, 8, 6, 3, 1, 9]
```

**Dificultades**

Combinación del backtraking con heuristica

Manejo de multiples soluciones

Comparación de rendimiento

**Conclusiones**

El software implementa tres enfoques distintos de backtracking para resolver Sudokus, lo que permite comparar su desempeño y apreciar la importancia de aplicar heurísticas como MRV y forward checking en problemas de búsqueda. Los resultados muestran que, aunque el algoritmo básico es funcional, las versiones con heurísticas mejoran significativamente la eficiencia en términos de intentos y tiempo. Sin embargo, el diseño puede fortalecerse con una estandarización en los valores de retorno, mejor manejo de variables globales, validación previa del tablero y pruebas automatizadas. En conclusión,

el programa cumple su propósito académico y didáctico, pero requiere ajustes en la estructura y robustez para alcanzar un nivel más profesional y escalable.

**Referencias**

https://www.geeksforgeeks.org/sudoku-backtracking-7/