




25 DE NOVIEMBRE DE 2024

SUDOKU
PROYECTO FINAL

FREDA MARIA PEREZ NOVELO
EMMANUEL MORALES SAAVEDRA
Técnicas Algorítmicas



Justificación de la Técnica

Elegí implementar un enfoque de *algoritmos voraces con retroceso (backtracking)* debido a las características específicas del problema del Sudoku. Este enfoque es particularmente adecuado porque permite explorar soluciones de manera sistemática, probando cada posible valor para una celda vacía mientras verifica las restricciones impuestas por las reglas del Sudoku: no repetir números en filas, columnas ni en subcuadrantes de 3x3.

El retroceso asegura que, si una elección no conduce a una solución válida, el algoritmo puede retroceder al último estado válido y probar una alternativa. Este enfoque, aunque basado en fuerza bruta, se optimiza mediante validaciones locales en cada paso, lo que reduce significativamente el espacio de búsqueda y mejora la eficiencia.

Alternativamente, métodos como *Divide y Vencerás* y *Programación Dinámica* no se adaptan tan bien al problema del Sudoku. Divide y Vencerás no puede dividir el tablero en subproblemas independientes debido a las dependencias globales (restricciones entre filas, columnas y subcuadrantes). La Programación Dinámica requiere solapamiento entre subproblemas, lo cual no es evidente en este caso. Por lo tanto, *backtracking* surge como la opción más intuitiva y práctica.

Complejidad Computacional

1. Complejidad Temporal:

- En el peor de los casos, la complejidad temporal del algoritmo es $O(9^N)$, donde N es el número de celdas vacías. Cada celda vacía puede contener uno de los nueve valores posibles, y el algoritmo explora todas las combinaciones posibles para encontrar una solución. Sin embargo, gracias a las validaciones realizadas en cada paso (verificación de filas, columnas y subcuadrantes), se descartan rápidamente combinaciones inválidas, reduciendo el número de opciones que el algoritmo necesita explorar en la práctica.

2. Complejidad Espacial:

- La complejidad espacial del algoritmo es $O(N)$, donde N corresponde al número de celdas vacías, ya que la pila de recursión almacena el estado del tablero en cada nivel del árbol de decisiones.

3. Tiempo de Ejecución Observado:

- Durante la ejecución del programa en un Sudoku estándar (9x9) de dificultad moderada, el tiempo de solución fue de menos de 100 ms en una máquina con procesador moderno. Para Sudokus más complejos, el tiempo de ejecución podría aumentar, pero sigue siendo aceptable gracias a las optimizaciones intrínsecas del algoritmo.

Resultados Obtenidos

El algoritmo se probó en un tablero inicial de Sudoku con varias celdas vacías. Los resultados fueron los siguientes:

- **Sudoku inicial:**

	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

- **Sudoku resuelto:**

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

El tiempo de ejecución fue de **85 ms**, y la solución generada cumple con todas las restricciones del juego.

Comparaciones con Otras Técnicas

1. Divide y Vencerás:

- Este enfoque no es efectivo para Sudokus porque el problema no puede dividirse fácilmente en subproblemas independientes. Cada subproblema dependería de los valores en otras celdas, lo que complica la combinación de soluciones parciales.

2. Programación Dinámica:

- Aunque poderosa para problemas con subproblemas solapados, la Programación Dinámica no es adecuada aquí porque cada celda del Sudoku depende de restricciones globales. Además, la implementación sería más compleja que el enfoque de retroceso.

3. Algoritmos Voraces sin Retroceso:

- Un enfoque completamente voraz intentaría llenar cada celda basándose únicamente en la mejor decisión local posible. Sin embargo, esto no garantiza una solución correcta para Sudokus con múltiples dependencias y restricciones globales.

Conclusión

El uso de *algoritmos voraces con retroceso* resultó ser una solución eficiente y práctica para resolver el Sudoku. Este enfoque es ideal para problemas con restricciones globales que requieren exploración sistemática del espacio de soluciones. El algoritmo mostró un rendimiento excelente en términos de tiempo de ejecución y calidad de la solución, siendo fácil de implementar y comprender.

Este ejercicio demostró la importancia de seleccionar la técnica algorítmica adecuada según las características del problema, optimizando tanto la complejidad computacional como la facilidad de implementación.