Solución de juego Sudoku mediante agente de inteligencia artificial

Introducción a la inteligencia artificial

1° Estevan Garcia Niño
Dept. de Matematicas
Universidad Nacional de Colombia
Bogota, Colombia
dgarcian@unal.edu.co

2° Jaime Macias Sanchez
Dept. de Ingeniería
Universidad Nacional de Colombia
Bogota, Colombia
jmaciass@unal.edu.co

3° Santiago Tovar Mosquera
Dept. de Matematicas
Universidad Nacional de Colombia
Bogota, Colombia
satovarmo@unal.edu.co

***Resumen*—El juego Sudoku, gracias a su naturaleza, puede ser solucionado por algoritmos de inteligencia artificial al ser planteado como *Constraint Satisfaction Problem* (CSP). Presentar una aplicación que solucione Sudokus brinda beneficios a la humanidad en el área de la salud, entretenimiento, y es base para investigaciones más profundas en biología. En este trabajo se desarrolló un aplicativo capaz de solucionar Sudokus válidos dados por el usuario o generados aleatoriamente. Se usó el algoritmo CSP con la siguiente configuración particular: búsqueda backtracking, tres tipos de inferencia y selección de siguientes variables por *Minimum-remaining-values* (MRV). El algoritmo no presenta problemas de desempeño, pero para el rendimiento computacional es recomendable aprovechar los beneficios de las inferencias sencillas como *Arc Consistency* (FC). Aunque el aplicativo cumple la función, es recomendable hacer más comparaciones con otras configuraciones de CSP o directamente con otros algoritmos.**

***Palabras clave*—Sudoku, Agente de inteligencia artificial, Problema de satisfacción de restricciones**

I. Introducción

El Sudoku es un juego tipo *puzzle* que en sus inicios fue muy popular en Japón, y es en este país que recibió el nombre. Presentado en un tablero de 9x9 casillas, en el Sudoku se busca completar cada casilla con números del 1 al 9 sin que se repitan por filas, ni columnas, ni sub-celdas (casillas de 3x3). Un Sudoku válido tiene una única solución [Davis, 2006]. El problema de solucionar este juego viene dado por el estado inicial (celdas fijadas con números de antemano) y las tres restricciones ya comentadas.

La solución de Sudokus trae potenciales beneficios a la salud humana. Como se describe en [Ogawa et al., 2016], resolver Sudokus mejora el razonamiento gramatical, la memoria de trabajo espacial y facilita las tareas de memoria episódica. Por otro lado, resolver un problema de restricciones como este juego impone las bases para otras aplicaciones como el plegamiento de proteínas [Ercsey-Ravasz and Toroczkai, 2012], por lo cual debe tratarse con seriedad.

El principal problema de esta investigación es proveer a la sociedad en general de una aplicación simple y efectiva en la solución de Sudokus. Cualquier persona, sin conocimientos específicos, que haga uso de la aplicación podrá entender, plantear y solucionar Sudokus. Podrá usarse en periódicos, revistas de pasatiempos o en hospitales que estén planteando nuevas terapias o ensayos clínicos. En específico, se busca desarrollar un software que admita Sudokus, los solucione y presente en una interfaz de usuario amigable.

La literatura nos presenta diferentes formas de solucionar Sudokus con técnicas de inferencia de satisfactibilidad proposicional en tiempo polinomial (SAT por sus siglas en inglés) [Lynce and Ouaknine, 2006], por medio de metaheurísticas [Lewis, 2007], gracias al algoritmo de búsqueda de armonía también se encuentra solución [Geem, 2007]. Pero también se pueden plantear algoritmos genéticos [Mantere and Koljonen, 2007].

Como se ha mencionado, la resolución de un Sudoku es un problema de restricciones y gracias a [Delahaye, 2006] sabemos que también es un problema matemático derivado del *cuadrado latino*. Además, el Sudoku es un juego creado especialmente para entretención de personas. Por estas razones, es lógico solucionar el problema por medio de técnicas de inteligencia artificial, específicamente CSP (constraint satisfaction problem) con los algoritmos presentados en [Russell and Norvig, 2010].

II. Objetivos

Implementar de manera correcta el problema de resolución de Sudokus como un CSP evaluando qué algoritmos son más óptimos y eficientes para el propósito en cuestión.

II-A. Objetivos específicos

- Lograr la correcta abstracción y representación del problema en Python.
- Desarrollar de manera eficiente los algoritmos que ayudan a la resolución de un CSP y comparar sus resultados.
- Enfocarnos principalmente en el algoritmo de backtracking, pues según se ve en [Russell and Norvig, 2010] es el algoritmo idóneo para el problema en cuestión, explotando al máximo su capacidad para evaluarlo comparativamente de forma adecuada.

III. Materiales y métodos

III-A. Agente de inteligencia artificial

Utilizamos el marco de trabajo propio de los agentes de inteligencia artificial. En consecuencia, planteamos la estructura del agente de la siguiente manera:

- **PEAS** (Performance Measure, Environment, Actuators, Sensors): El agente debe poder encontrar o no una solución al Sudoku planteado por el usuario, luego la medida de desempeño es una frecuencia de booleanos (consigue una solución – no consigue una solución). El entorno del agente se reduce al tablero con valores iniciales que proporciona el usuario. El agente podrá actuar (presentar los resultados) mediante la pantalla del computador. Por último, el sensor es la lectura de valores en las respectivas casillas del sudoku.
- **Propiedades del entorno de tareas**: Se está tratando con un entorno completamente observable, de agente único, determinístico, secuencial, estático, discreto y conocido.
- **Arquitectura**: El agente puede correr sobre un computador convencional que pueda ejecutar código Python.
- **Programa**: Basado en modelos.
- **Algoritmo de resolución**: CSP con las características:
  - Conjunto de variables $\{X_1, X_2, \ldots, X_n\}$ que representan las casillas del Sudoku.
  - Un dominio para cada una de las variables $\{D_1, D_2, \ldots, D_n\}$. En este caso, para casillas vacías cada dominio es igual $D_i = \{1, 2, \ldots, 9\}$. Las casillas llenan tienen un dominio singleton que consiste en el número con el que vienen rellenas.
  - El conjunto de restricciones. En nuestro caso es de 27 elementos (9 restricciones por cada sub-celda del Sudoku, 9 para las lineas horizontales y 9 para las verticales).

  La solución del problema incluye búsqueda *backtracking*; con inferencia *FC* (Arc Consistency), inferencia *MAC* (Maintaining Arc Consistency) o sin inferencia; selección de variable por *MRV* (minimum-remaining-values).

Con esto en mente, se tiene el diagrama del agente en la Figura 1.

III-B. Configuración experimental

La solución propuesta se evaluó en dos configuraciones experimentales:

- Evaluación de desempeño: Se proporcionó varios Sudokus con diferentes cantidades de casillas llenas, evaluando por cada estado inicial cuánto tardaba el programa en mostrar una solución a la configuración del juego. Con este experimento evidenciamos en qué condiciones el agente logra proporcionar una respuesta acertada.
- Evaluación computacional: Con una configuración de Sudoku dada se evaluó cuánto tiempo necesita el algoritmo para completar la tarea. Esto se logró con un temporizador dentro de la ejecución de la aplicación.

El algoritmo base para los experimentos esta dado por el más simple, CSP sin inferencia.

IV. Resultados

Consideremos el que es para muchos "el Sudoku más difícil del mundo", el cual podemos ver solucionado en la Figura 2, este sera el Sudoku que usaremos tanto para la evaluación de desempeño como para la evaluación computacional.

Para el desarrollo de este proceso, se construyó un archivo `Experimentos.py` (dentro del repositorio del aplicativo puede explorar todos los archivos, https://github.com/EstevanGN/Sudoku), en donde inicializamos el Sudoku descrito con anterioridad. En la Figura 2 podemos ver claramente los números que hacen parte de la solución al Sudoku; es por ello que adicionalmente creamos un arreglo en donde guardamos las entradas de las casillas que deberían estar vacías en un principio.

Con base en ese arreglo, podemos hacer una elección aleatoria para ir vaciando n casillas del Sudoku y así realizar nuestra evaluación de desempeño.

Realizamos 10 iteraciones, en cada una tomamos el arreglo de entradas y elegimos 6,12,18,... (aumentando en 6) parejas para ir volviendo dichas entradas vacías en el Sudoku y comparar los resultados del mismo. Para este caso usamos una resolución para CSP sin inferencia mediante el algoritmo de backtracking, y obtuvimos una gráfica como la que podemos ver en la Figura 3.

Por otra parte, para nuestro otro experimento de evaluación computacional usamos 3 tipos de inferencias: El primero es no contar con ninguna inferencia, el segundo caso es Forward Chaining y finalmente Maintaining Arc Consistency. Cada una de estas se corrió 10 veces intentando resolver el Sudoku más difícil del mundo (La versión con 21 casillas llenas). Aquí promediamos los tiempos de cada inferencia y mostramos el resultado en la Figura 4.

V. Discusión

Como se esperaba, la aplicación es capaz de encontrar soluciones siempre y cuando el Sudoku sea válido. Comentarios adicionales sobre el desempeño vienen dados por la Figura 3, el algoritmo requiere mayor tiempo para encontrar una solución cuando la cantidad de casillas es cercana a 21; es decir el aso del Sudoku original. Es en este punto donde la cantidad de restricciones dificulta la tarea.

Es evidente además que mientras se tenga un número alto de casillas llenas, el tiempo de búsqueda sea extremadamente corto, pues los dominios de las variables se restringen lo cual facilita la obtención de resultados.

Así mismo, es razonable que al disminuir la cantidad de casillas llenas el tiempo de computo sea mucho menor. Es muy posible que no se tenga una única solución, lo que permite al sistema hallar una de tantas posibles en un tiempo extremadamente corto.

Respecto al rendimiento computacional, es claro que para el problema en cuestión FC tiene un mejor desempeño respecto a las demás configuraciones para inferencia. La sencillez de la inferencia FC provee al algoritmo un mayor rendimiento, como en contraposición una inferencia muy compleja (MAC) añade complicaciones extra. Es decir, para nuestro problema

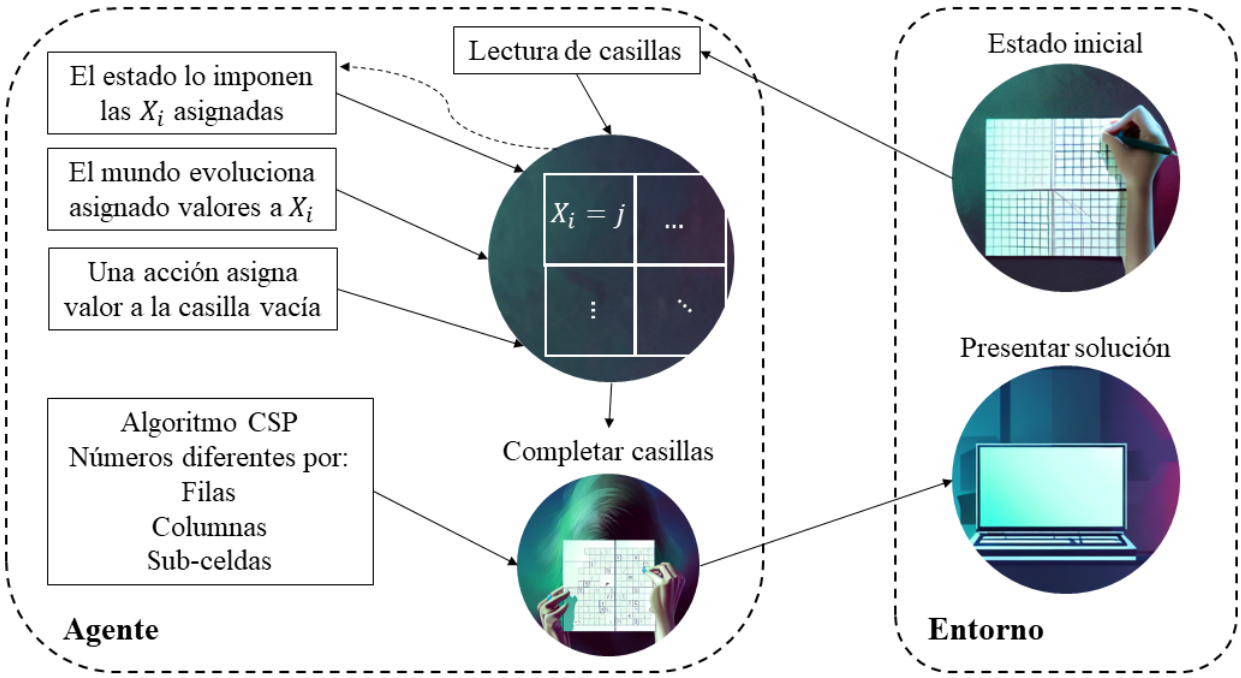

Figura 1. Modelo del agente de inteligencia artificial diseñado para solucionar Sudokus. Imágenes desarolladas en https://hotpot.ai/art-generator.

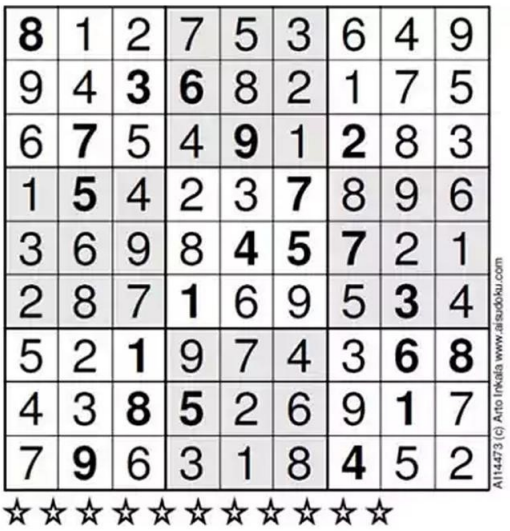

Figura 2. El Sudoku mas difícil del mundo. Recuperado de https://www.lt10.com.ar/noticia/226499--un-matematico-creo-el-sudoku-mas-dificil-del-mundo#:~:text=Se%20llama%20Everest%20y%20fue,lista%20sin%20mirar%20la%20soluci%C3%B3n%3F&text=Las%20reglas%20no%20son%20muchas%20y%20son%20muy%20sencillas.

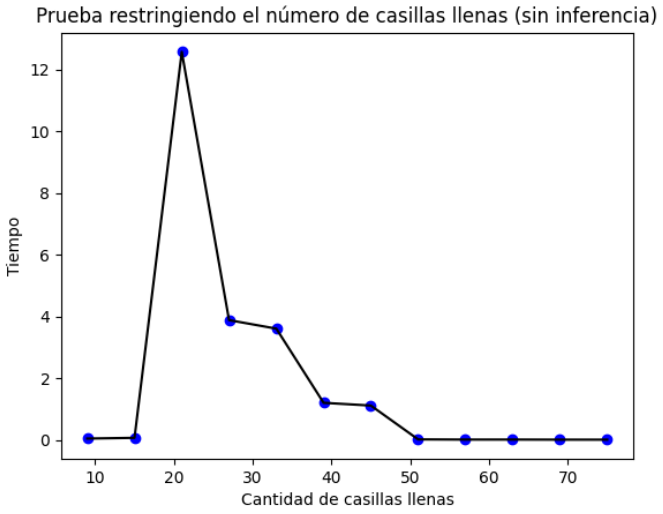

Figura 3. Prueba restringiendo el número de casillas llenas y observando el tiempo que tarda el algoritmo en encontrar una solución.

en donde el algoritmo siempre encuentra la solución, se puede aprovechar el añadido de las inferencias pero sin incurrir en aquellas muy complejas.

A pesar de obtener resultados (soluciones) para el problema del Sudoku, este proyecto se basa en un abordaje puntual por CSP con diferentes configuraciones de inferencia. En futuras aproximaciones es recomendable explorar configuraciones añadidas en el tipo de búsqueda (no solo backtracking) y en la forma de escoger la siguiente variable (no solo MRV).

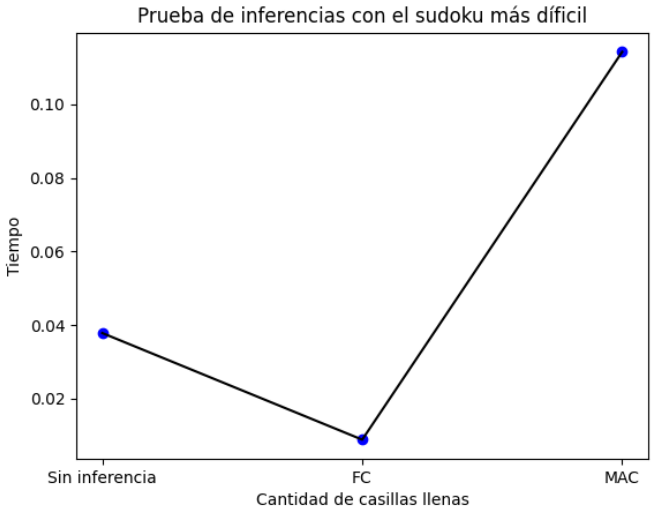

Figura 4. Prueba de rendimiento computacional para diferentes inferencias con "el Sudoku más difícil".

Por otro lado, también es importante abordar otras metodologías de inteligencia artificial. En [Russell and Norvig, 2010] se sugieren las *búsquedas locales* para los problemas de CSP, sin embargo no parecen ser una excelente idea dada la cantidad de variables y restricciones en el Sudoku. Explorar algoritmos de búsqueda *primero en amplitud* o *de profundidad limitada*, como se presentan en [Lina and Rumetna, 2021], son opciones de solución para el puzzle. Otra alternativa de inteligencia artificial para Sudokus es el algoritmo *β-Hill Climbing*, que en [Al-Betar et al., 2017] muestra buenos resultados. Por último, una aproximación por *Deep Learning* similar a [Vamsi et al., 2021] nos brindaría un nuevo punto de vista.

En general, con las posibilidades enunciadas en el párrafo anterior, un futuro trabajo buscaría evaluar y comparar desempeño y consumo computacional entre las soluciones. Además de mejorar la interfaz gráfica con ayuda de un diseñador, esto en busca de aumentar la experiencia del usuario y hacer una aplicación de producción.

VI. Conclusiones

El algoritmo de CSP para la solución de Sudokus es una excelente herramienta. Siempre se puede encontrar una solución cuando el Sudoku es válido. Además, si se busca un mejor rendimiento es recomendable usar los beneficios de las inferencias sencillas como FC.

Para este caso en particular, inferencias con complejidades mayores no aportan gran ganancia al aplicativo, por el contrario reducen su rendimiento. También es importante recalcar que no es necesario usar inferencia para poder encontrar solución en la aplicación, y esta escogencia no afecta de sobremanera el rendimiento general.

La aplicación dada en el repositorio https://github.com/EstevanGN/Sudoku cumple con la presentación de una solución para cualquier Sudoku dado. Tiene una interfaz sencilla e intuitiva para el usuario.

Referencias

[Al-Betar et al., 2017] Al-Betar, M. A., Awadallah, M. A., Bolaji, A. L., and Alijla, B. O. (2017). β-hill climbing algorithm for sudoku game. In *2017 Palestinian International Conference on Information and Communication Technology (PICICT)*, pages 84–88. IEEE.

[Davis, 2006] Davis, T. (2006). The mathematics of sudoku.

[Delahaye, 2006] Delahaye, J.-P. (2006). The science behind sudoku. *Scientific American*, 294(6):80–87.

[Ercsey-Ravasz and Toroczkai, 2012] Ercsey-Ravasz, M. and Toroczkai, Z. (2012). The chaos within sudoku. *Scientific reports*, 2(1):1–8.

[Geem, 2007] Geem, Z. W. (2007). Harmony search algorithm for solving sudoku. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 371–378. Springer.

[Lewis, 2007] Lewis, R. (2007). Metaheuristics can solve sudoku puzzles. *Journal of heuristics*, 13(4):387–401.

[Lina and Rumetna, 2021] Lina, T. N. and Rumetna, M. S. (2021). Comparison analysis of breadth first search and depth limited search algorithms in sudoku game. *Bulletin of Computer Science and Electrical Engineering*, 2(2):74–83.

[Lynce and Ouaknine, 2006] Lynce, I. and Ouaknine, J. (2006). Sudoku as a sat problem. In *AI&M*.

[Mantere and Koljonen, 2007] Mantere, T. and Koljonen, J. (2007). Solving, rating and generating sudoku puzzles with ga. In *2007 IEEE congress on evolutionary computation*, pages 1382–1389. IEEE.

[Ogawa et al., 2016] Ogawa, E. F., You, T., and Leveille, S. G. (2016). Potential benefits of exergaming for cognition and dual-task function in older adults: a systematic review. *Journal of aging and physical activity*, 24(2):332–336.

[Russell and Norvig, 2010] Russell, S. and Norvig, P. (2010). *Artificial intelligence a modern approach*. Pearson Education, Inc.

[Vamsi et al., 2021] Vamsi, K. S., Gangadharabhotla, S., and Sai, V. S. H. (2021). A deep learning approach to solve sudoku puzzle. In *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 1175–1179. IEEE.