Perspectivas del Sudoku en el Ámbito del Razonamiento

Denisa Alexandru
Máster Universitario en Inteligencia Artificial
Universidad Politécnica de Madrid
Madrid, España
denisa.alexandru@alumnos.upm.es

Miguel Barragán
Máster Universitario en Inteligencia Artificial
Universidad Politécnica de Madrid
Madrid, España
miguel.barragan.rodriguez@alumnos.upm.es

***Abstract*—Los sudokus son un tipo de puzzle común en la sociedad y que a día de hoy siguen siendo jugados. Hay bastanes estrategias que se utilizan para poder resolverlos, en especial los sudokus de alta dificultad, todas estas estrategias que hemos ido aprendiendo a medida que se iban resolviendo sudokus son las que hemos querido plasmar en un algoritmo. El objetivo de este trabajo es crear un algoritmo basado en reglas que replicara las estrategias desarrolladas por los seres humanos y que permita entender cómo aplicarlas para resolver un sudoku. Además la compararemos con las técnicas más usadas para resolverlos actualmente como puedan ser los algoritmos basados en restricciones.**

***Index Terms*—sudoku, algoritmos, reglas, restricciones**

I. Introducción

La naturaleza lógica del Sudoku ha impulsado una extensa investigación para desarrollar algoritmos de resolución eficientes. La gran cantidad de posibles configuraciones, con aproximadamente $7*10^{21}$ cuadrículas de Sudoku distintas solo para los Sudokus Clásicos, presenta un desafío formidable para los algoritmos de búsqueda convencionales, como el backtracking ingenuo. A pesar de sus reglas aparentemente sencillas, resolver Sudokus sigue siendo una tarea compleja tanto para humanos como para computadoras [1].

Este trabajo explora los algoritmos de resolución de Sudoku, no solo desde la perspectiva de la eficiencia, sino con un enfoque en la solidez lógica y las metodologías comprensibles para los humanos. El objetivo principal no es optimizar el proceso de solución, sino desentrañar las bases lógicas de una manera accesible a la intuición humana. La pregunta central de este trabajo es si es posible construir un algoritmo lógico transparente que resuelva eficazmente Sudokus usando estrategias humanas.

Este trabajo se basa en intersección entre el razonamiento lógico, las estrategias humanas comúnmente usadas por los aficionados al Sudoku y la eficiencia computacional. Aunque se han propuesto numerosos algoritmos, muchos carecen de una justificación clara. El algoritmo propuesto busca romper con esta tendencia al proporcionar una explicación sistemática y comprensible de los pasos que realiza. Situado en el ámbito de la Inteligencia Artificial Explicativa (XAI), el algoritmo tiene como objetivo no solo ser un solucionador de Sudoku, sino también una herramienta para explicar el proceso de razonamiento detrás de cada movimiento.

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

Fig. 1. Tablero de Sudoku, con las filas, columnas y bloques resaltados en amarillo

II. Bases del Sudoku

El Sudoku se basa en una cuadrícula de 9x9 espacios, donde el tablero se compone de casillas dispuestas en filas y columnas, configurando un total de 81 espacios (Figura 1).

Únicamente se pueden utilizar números del 1 al 9 para completar las casillas. Esta restricción agrega complejidad y requiere una cuidadosa consideración al colocar cada número.

La cuadrícula se divide en bloques 3x3, columnas verticales y filas horizontales, cada una con la restricción de que un número determinado solo puede aparecer una vez. Es decir, ningún número se repite en un mismo bloque, columna o fila.

La resolución del Sudoku culmina con la correcta disposición de los números en la cuadrícula. Una vez que cada número del 1 al 9 ocupa su lugar sin violar las reglas establecidas, el juego se completa exitosamente [2].

A. Soluciones del Sudoku

Un aspecto relevante a tener en cuenta a la hora de la resolución de los Sudokus es el número de soluciones que tiene un Sudoku.

La respuesta simplificada sería que el Sudoku sólo tiene una solución. Sin embargo, para que esto sea cierto se necesitan cumplir varias condiciones.

En realidad el número de soluciones depende del número de pistas inciales y su posición.

Determinar el número mínimo de pistas necesario para garantizar una solución única en un rompecabezas de Sudoku. En un estudio realizado por McGuire *et al.* [3] mediante una extensa exploración informática, se establece que el mínimo número de pistas es de 17. Esto además, se confirma con el hecho de que no existe ningún Sudoku que tenga una única solución con 16 pistas. En el estudio mencionado, se utiliza un algoritmo para enumerar y encontrar subconjuntos inevitables, mediante una exploración exhaustiva de cada posible cuadrícula de solución de Sudoku. La búsqueda implicó analizar individualmente cada cuadrícula de Sudoku completa, buscando sistemáticamente un subconjunto de 16 pistas cuya única completitud correspondiera a la cuadrícula dada. Los resultados revelaron la necesidad imperante de al menos 17 pistas para garantizar una solución única.

El concepto de subconjunto inevitable, que se refiere a la disposición específica de pistas en la cuadrícula de Sudoku que puede reorganizarse para producir una cuadrícula diferente, es esencial para llegar a cuadrículas con una única solución. La investigación demuestra que si un conjunto de pistas no cubre todos los subconjuntos inevitables, esa cuadrícula no puede ser utilizada como conjunto de pistas iniciales. Esto se debe a que si un conjunto inevitable no se queda relleno, se podrían tener distintas soluciones, lo que haría que la cuadrícula quede inválida.

En términos más sencillos, para asegurar una solución única en un rompecabezas de Sudoku, se debe incluir al menos una pista de cada subconjunto inevitable.

Para aclarar este concepto, se va a explicar mejor utilizando la cuadrícula de la Figura 2. Nos centraremos en el subconjunto inevitable representado con celdas color gris claro. Se puede observar que en las 4 celdas representadas, el 5 y el 9 se repiten. La solución inicial tiene el 9 y el 5 situados en el bloque de la izquierda, y el y el 9 situados en el bloque central. sin embargo, si se fueran a quitar estos número, dejando que el usuario complete la cuadrícula se generaría ambigüedad, ya que se podría generar la solución inicial, o se podría generar una nueva solución en la que situaríamos en el bloque de la izquiera primero el número 5 en la celda de arriba y el número 9 en la celda de abajo, y en le bloque central lo contrario. Esto daría lugar a dos posibles soluciones, lo que invalidaría esta cuadrícula.

Sin embargo, los Sudokus utilizados actualmente, cumplen con las dos condiciones antes mencionadas: siempre tienen más de 17 pistas, y además hay al menos una pista en cada subconjunto inevitable. Por lo tanto, estos Sudokus sólo tienen una solución, y es esto lo que se considerará como base en nuestro trabajo.

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

Fig. 2. Cuadrícula de Sudoku, en la que están representados dos subconjuntos inevitables en gris claro y en gris oscuro.

III. Enfoque lógico

A. Razonamiento

El razonamiento, como proceso cognitivo fundamental, se erige como una actividad mental sistemática que implica el uso de la lógica y el pensamiento crítico para derivar conclusiones o tomar decisiones fundamentadas. Este proceso implica analizar información, evaluar evidencia y realizar inferencias paso a paso, con el objetivo de llegar a resultados coherentes y justificados. Es esencial considerar hechos, establecer conexiones lógicas y seguir un enfoque metódico para lograr conclusiones informadas[4].

Dentro del amplio espectro del razonamiento, se distinguen varios tipos, cada uno con métodos particulares para derivar conclusiones basadas en diversas formas de evidencia y principios lógicos. Entre ellos, destacan el razonamiento deductivo (se obtienen conclusiones particulares a partir de unas reglas generales), inductivo (se deduce una regla general a partir de casos particulares), abductivo (razonamiento en el cual se formulan y evalúan hipótesis explicativas [5]) y analógico (identificación de similitudes o analogías entre situaciones, objetos o conceptos diferentes), cada uno aportando una perspectiva única al proceso de pensamiento (Figura 3) [4].

Fig. 3. Tipos de razonamiento

El razonamiento deductivo, un componente central del proceso cognitivo, implica la realización de inferencias lógicas. Es una forma de razonamiento que ocupa un lugar destacado en la inteligencia, siendo evaluado en pruebas de inteligencia que incluyen problemas en razonamiento deductivo [6].

Todos aquellos dotados de capacidad mental reconocen que ciertas inferencias son válidas porque no existen contraejemplos, es decir, no hay posibilidades en las cuales las premisas sean ciertas pero la conclusión no lo sea. Esta noción subyace en el razonamiento deductivo [6].

Un conjunto de reglas formales constituye un sistema para demostrar que las conclusiones pueden derivarse de las premisas, y son sensibles a la forma lógica de las premisas, especificada por la gramática. Los lógicos han demostrado

que un sistema de deducción natural para el cálculo proposicional tiene la propiedad deseable de que cualquier inferencia que pueda demostrarse usando las reglas formales es válida según la semántica, y la propiedad adicional deseable de que cualquier inferencia que es válida según la semántica es demostrable usando las reglas formales[6].

El Sudoku puede considerarse un problema de razonamiento deductivo debido a su naturaleza basada en reglas lógicas y la necesidad de inferir conclusiones de información dada. En el Sudoku, las reglas preestablecidas gobiernan la disposición de los números en la cuadrícula, y el jugador debe utilizar un proceso deductivo para deducir la ubicación de los números restantes.

B. Adivinanza

El razonamiento se contrapone a menudo con la intuición, una forma más instintiva y espontánea de toma de decisiones fundamentada en estados internos, sentimientos o pensamiento asociativo rápido. Mientras que la intuición se apoya en corazonadas o respuestas inmediatas, el razonamiento exige un análisis consciente y deliberado de la información para llegar a conclusiones bien fundamentadas [4]. En este contexto surge el concepto de adivinanza.

El concepto de ”adivinanza” en los rompecabezas de Sudoku se define como la acción de ”colocar un dígito en una celda sin razonamiento lógico”. Aunque su definición puede ser objeto de debate, implica una reflexión sobre cómo realizar suposiciones y si estas pueden ser tácticas. En situaciones donde las estrategias lógicas han agotado sus posibilidades, surge la opción de realizar una suposición educada, llenando una celda con un dígito permitido para evaluar si desde ese punto se puede generar una solución [1].

A pesar de que esta táctica puede fallar, no se considera una pérdida total, ya que aporta conocimiento valioso: descarta la posibilidad de que el dígito en esa celda sea parte de la solución. Este enfoque, característico de muchos algoritmos de retroceso como el *backtracking* del que se hablará en futuras secciones, refleja el proceso lógico que una persona seguiría al enfrentarse a un callejón sin salida en un rompecabezas de Sudoku, fusionando la necesidad de razonamiento deductivo con estrategias más intuitivas cuando el escenario lo demanda[1].

IV. Problemas de Satisfacción de Restricciones

A. Definición General

Para problemas complejos como es el Sudoku, se puede usar una representación factorizada para cada estado, la cual se materializa como un conjunto de variables, cada una con su asignación única de valor. El éxito se alcanza cuando cada variable exhibe un valor que cumple con las restricciones específicas impuestas sobre ella. Este enfoque detallado para describir un problema da origen a lo que comúnmente se denomina un Problema de Satisfacción de Restricciones (CSP).

En esencia, el CSP encapsula la noción de encontrar una combinación de valores para las variables involucradas que satisfaga de manera simultánea todas las restricciones establecidas. Esta formulación permite abordar problemas diversos donde múltiples condiciones deben cumplirse para alcanzar una solución coherente.

Los algoritmos que empleamos para atravesar CSPs aprovechan la estructura inherente de los estados, utilizando heurísticas de propósito general en lugar de específicas del problema. La esencia radica en eliminar estratégicamente vastas extensiones del espacio de búsqueda al identificar pares variable-valor que transgreden las restricciones.

Un CSP se compone de tres componentes fundamentales: X, D y C:

- **X** constituye un conjunto de variables, denotado como $\{$X1, ..., Xn$\}$.
- **D** es una colección de dominios, $\{$D1, ..., Dn$\}$, donde cada dominio corresponde a una variable.
- **C** es un conjunto de restricciones que delinean combinaciones permisibles de valores.

Cada dominio Di encapsula un conjunto de valores permitidos, $\{$v1, ..., vk$\}$, para la variable Xi. Cada restricción Ci se define mediante un par (alcance, rel), donde el alcance es una tupla de variables participantes y rel es una relación que especifica los valores permitidos para esas variables. Una relación puede manifestarse como una lista explícita de tuplas que satisfacen la restricción o como una entidad abstracta que admite operaciones como comprobar la membresía de una tupla y enumerar los miembros de la relación. Por ejemplo, si X1 y X2 tienen ambos el dominio A, B, la restricción que exige valores distintos se puede expresar como (X1, X2), [(A, B), (B, A)] o como (X1, X2), X1 = X2.

La solución a un CSP se delinea mediante la definición de un espacio de estados y la conceptualización de una solución. Cada estado de ASIGNACIÓN en un CSP se caracteriza por la asignación de valores a algunas o todas las variables $\{$Xi = vi, Xj = vj, ...$\}$. Una asignación libre de violaciones de restricciones se denomina consistente o legal. Una asignación completa abarca cada variable, y una SOLUCIÓN DE ASIGNACIÓN COMPLETA para un CSP es una asignación completa y consistente. Por el contrario, una asignación parcial se refiere a valores asignados solo a algunas variables.

Las restricciones en los CSPs se clasifican en dos categorías: restricciones binarias y restricciones globales. Las restricciones binarias involucran únicamente dos variables. Este tipo de restricciones constituyen un subconjunto específico de los CSPs y se representan visualmente mediante un grafo de restricciones, donde los nodos son variables y los arcos reflejan las relaciones binarias entre ellas.

En contraposición, las restricciones globales pueden implicar un número arbitrario de variables. Un ejemplo común de restricción global es Alldiff, que exige que todas las variables involucradas posean valores distintos. En el contexto de problemas de Sudoku, esta restricción se aplica a cada fila o columna, garantizando que cada casilla contenga un valor único.

Sin embargo, los CSPs se enfrentan al problema de la inconsistencia en restricciones Alldiff. Un método simple para detectar inconsistencias sugiere que si m variables participan

en una restricción y tienen en conjunto n posibles valores distintos, donde $m > n$, entonces la restricción es inherentemente insatisfactoria.

Una manera de solucionar estos desafíos es mediante el algoritmo AC-3, conocido como Arc-Consistency 3. Se utiliza especialmente cuando se trata de mantener la consistencia de las restricciones en relaciones binarias. Este algoritmo se distingue por su capacidad para abordar problemas complejos mediante la aplicación estratégica de restricciones, garantizando la coherencia en las relaciones entre variables y facilitando una navegación precisa a través de los espacios de solución en los CSPs.

La consistencia de arcos implica asegurar que, para cada restricción binaria, los valores en el dominio de una variable sean coherentes con los valores en el dominio de la variable relacionada. En el contexto de un rompecabezas de Sudoku, comenzamos expandiendo restricciones globales, como Alldiff, en restricciones binarias. Esto posibilita la aplicación directa del algoritmo AC-3.

El algoritmo AC-3 realiza iteraciones sucesivas. En cada iteración, se examinan las restricciones binarias, y los dominios de las variables se ajustan para garantizar la consistencia. Este ciclo continúa hasta que los dominios de todas las variables se reducen a conjuntos de valores singulares, indicando la consecución de una solución coherente para el CSP [7].

B. Sudoku como problema de satisfacción de restricciones

El Sudoku es formalmente considerado un Problema de Satisfacción de Restricciones (CSP), un enunciado matemático que modela situaciones en las que un conjunto de variables debe cumplir con ciertas restricciones para encontrar una solución. En el contexto del Sudoku, este rompecabezas se plantea de la siguiente manera:

- **Variables**: El tablero de Sudoku consta de 81 casillas dispuestas en una cuadrícula de 9x9. Cada casilla representa una variable del CSP. Utilizamos nombres de variables como A1, A2, ..., I8, I9 para identificar cada posición única.
- **Dominios**: Las casillas vacías tienen un dominio de valores posibles, generalmente 1, 2, 3, 4, 5, 6, 7, 8, 9, ya que se busca colocar un número del 1 al 9 en cada casilla. Las casillas prellenadas tienen un dominio constituido por un solo valor.
- **Restricciones**: Se establecen restricciones de tipo Alldiff para cada fila, columna y caja de 3x3. La restricción Alldiff garantiza que todos los valores en la misma fila, columna o caja sean distintos entre sí.

La formulación de este CSP implica encontrar asignaciones de valores a las variables (casillas) que cumplan con las restricciones Alldiff y, al mismo tiempo, completen adecuadamente el tablero. La solución al Sudoku, por lo tanto, es una asignación coherente y completa que satisface todas las restricciones establecidas.

Resolver el Sudoku se convierte, entonces, en una tarea de búsqueda y asignación de valores que cumplan con las reglas del juego. Los métodos de resolución de CSP, como el backtracking y la consistencia de arcos, son comúnmente utilizados para abordar eficientemente este desafío lógico [8].

En el contexto del Sudoku como un Problema de Satisfacción de Restricciones (CSP), el proceso de resolución implica aplicar estrategias específicas para satisfacer las condiciones del juego. Tomemos la variable E6 como ejemplo dentro de la Figura 4:

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

(a)

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

(b)

Fig. 4. (a) Cuadrícula del Sudoku (b) y sus soluciones

Al analizar las restricciones en la caja y columna correspondiente, se eliminan valores del dominio de E6 que contradicen las restricciones impuestas. Por ejemplo, si E6 está en la caja que tiene un 2 y un 8, se eliminan de su dominio no solo el 2 y 8 sino también otros valores inconsistentes como 1 y 7.

1) Aplicación del Algoritmo AC-3: Supongamos que las restricciones Alldiff se han expandido en restricciones binarias (como A1 = A2) para aplicar el algoritmo AC-3 directamente. Consideremos la variable E6, el cuadro vacío entre el 2 y el 8. Aplicando AC-3, eliminamos 2, 8, 1 y 7 de su dominio. Esto deja a E6 con un dominio de 4, es decir, conocemos la respuesta para E6. Aplicando AC-3 en otras variables como I6, se reducen sus dominios. La consistencia en la columna puede revelar valores únicos. Por ejemplo, si sabemos que E6 es 4, eliminamos otros valores de la columna, y finalmente, la consistencia de arcos puede inferir que A6 debe ser 1. Este proceso de consistencia de arcos se despliega a lo largo de todas las variables, y eventualmente, el algoritmo AC-3 puede resolver todo el rompecabezas. En nuestro problema, cada restricción Alldiff establece bijecciones entre variables y valores, y la consistencia de arcos es fundamental para reducir dominios y llegar a soluciones únicas [7].

C. Backtracking

El algoritmo de Backtracking es un enfoque fundamental utilizado en la resolución de Sudokus mediante la satisfacción de restricciones, especialmente al adoptar una estrategia de búsqueda en profundidad. El algoritmo opera seleccionando una variable a la vez, probando varios valores de su dominio hasta encontrar una solución. Si surge una inconsistencia, el algoritmo retrocede a la variable anterior, explorando valores alternativos [7].

Hay varios enfoques que mejoran este algoritmo:

1) Retroceso Cronológico: El retroceso cronológico implica volver al punto de decisión más reciente cuando una rama falla, ofreciendo una estrategia simplista. Sin embargo, enfoques de retroceso más inteligentes implican retroceder a una variable que podría rectificar el problema. Esto se logra mediante técnicas como el backjumping, que identifica un conjunto de asignaciones conflictivas para una variable específica. El algoritmo luego retrocede a la asignación más reciente en este conjunto de conflictos, evitando potencialmente iteraciones innecesarias [7].

2) Aprendizaje de Restricciones: En situaciones donde se alcanza una contradicción, el backjumping proporciona información sobre hasta dónde retroceder. Sin embargo, para evitar revisitar el mismo problema, se aplica el aprendizaje de restricciones. El aprendizaje de restricciones identifica un conjunto mínimo de variables del conjunto de conflictos responsables del problema. Este conjunto, conocido como un ”no-good”, se registra para mejorar la eficiencia de búsquedas futuras. Los solucionadores modernos de CSP aprovechan el aprendizaje de restricciones para optimizar el rendimiento en problemas intrincados [9].

3) Aplicación del backtracking al Sudoku: La aplicación del backtracking al Sudoku implica seleccionar un cuadro vacío y asignar valores sistemáticamente, retrocediendo cuando surgen conflictos. Aunque es efectivo para rompecabezas más pequeños, este método enfrenta desafíos en los más grandes debido al extenso espacio de búsqueda. Mejoras como la propagación de restricciones, la verificación anticipada y la heurística de valores mínimos restantes mitigan estos desafíos, haciendo que el retroceso sea más eficiente para rompecabezas más grandes [9].

4) Enfoque Híbrido: Mientras que el backtracking garantiza una solución para todos los rompecabezas de Sudoku válidos, su naturaleza de fuerza bruta puede ser intensiva computacionalmente. Existen enfoques híbridos, combinando el backtracking con mñetodos basados en estrategias humanas. El método híbrido incorpora reglas inteligentes y eliminación de candidatos para mejorar la eficiencia, proporcionando una solución para todos los rompecabezas de Sudoku válidos de manera más optimizada en comparación con el retroceso tradicional [10].

Sin embargo, el backtracking no permite obtener una explicabilidad de las reglas utilizadas a la hora de obtener una solución, de allí que nuestro enfoque se centre en el uso exclusivo de reglas humanas, aunque sea menos eficiente.

V. Resolución de Sudoku según Estrategias Humanas

A. Reglas

El algoritmo propuesto funciona en base a reglas. Una regla es una acción que se ejecuta cuando se cumplen ciertas condiciones, por ejemplo, si un microprocesador se calienta demasiado, la acción sería reducir la potencia para reducir la temperatura. En este aplicamos reglas al sudoku, el cuál normalmente se resuelve usando restricciones. Comparando ambas opciones, las restricciones son una mejor opción ya que es más eficiente y fácil de implementar, pero nuestro objetivo con este proyecto es traducir las estrategias que usamos los seres humanos cuando jugamos al sudoku y crear un algoritmo con esas estrategias, para esto la mejor forma es usando reglas. Nosotros proponemos 4 reglas las cuales son suficientes para resolver sudokus fáciles, medios y difíciles, pero los expertos y maestros no, para resolver estos se necesitarían bastantes más reglas y eso supera los límites para este proyecto. Antes de empezar a comentar las reglas hay que dejar claro unos términos. El algoritmo va celda a celda del sudoku resolviendo y para cada celda comprueba si puede aplicar una de estas reglas, la celda en la que esté actualmente comprobando la denominamos como **Celda Actual**. En el sudoku como no se pueden repetir números ni en filas, columas y cuadrantes, para cada celda hay una cantidad de números que no pueden existir, y, para los números que sí pueden existir los vamos a llamar **Números Posibles**.

1) Regla 1: Si la celda actual es la única celda restante por rellenar en una fila, columna o cuadrante, se rellena con el número que falte.

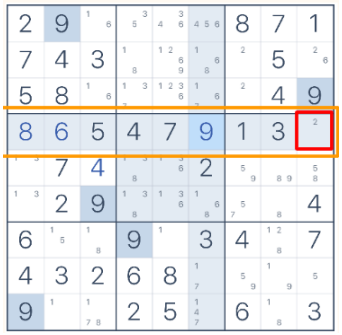

Fig. 5. Regla 1

En este ejemplo, podemos ver que la celda actual es la roja y el único número posible para esa celda es el 2 porque es la única celda restante de esa fila, así que se añadiría el número 2 en la celda actual.

2) Regla 2: Si en la celda actual se calculan los números posibles y sólo hay un candidato, entonces se añade ese número a la celda actual.

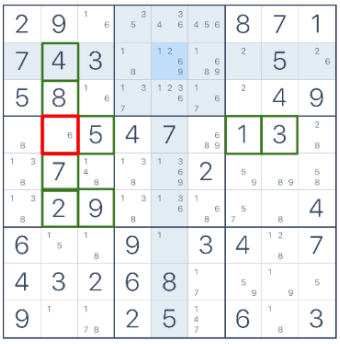

Fig. 6. Regla 2

En este ejemplo, podemos ver que el único número posible en la celda actual es el 6, ya que en su fila, columna y cuadrante ya están el resto de números.

3) Regla 3: Primero calculamos todos los números posibles de la celda actual, después hacemos lo mismo para todas las celdas en la misma fila. Después comprobamos si la celda actual tiene algún número posible que ninguna otra celda en la fila tenga, si es así, añadimos ese número a la celda. Si no lo encontramos en la fila, probamos con la columna y luego con el cuadrante.

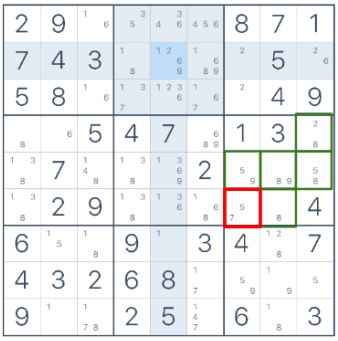

Fig. 7. Regla 3

En este ejemplo, podemos ver que en el cuadrante de la celda actual, tenemos calculados todos los números posibles, y podemos ver que la celda actual tiene como posible el número 7 y ninguna otra celda de su mismo cuadrante tiene como posible el número 7, por lo tanto añadimos el número 7.

4) Regla 4: Para la celda actual, se comprueban los cuadrantes que compartan su fila y su columna. Si todas las celdas vacías de alguno de estos cuadrantes pertenecen a la misa fila o columna que la celda actual, entonces se aplica esta regla, que lo que hace es quitar como números posibles en el resto de las celdas, los números que estén en el cuadrante con todas las celdas vacías en la misma fila o columna.

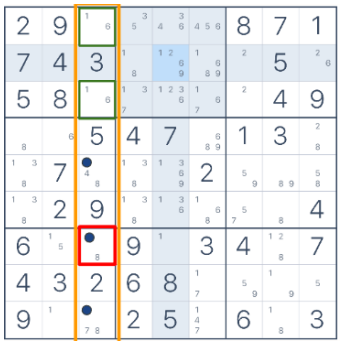

Fig. 8. Regla 4

En este ejemplo, podemos ver que en la tercera columna, el primer cuadrante tiene todas sus celdas vacías en la misma columna que la celda actual, por lo tanto los números posibles de ese cuadrante se quitan de el resto de celdas de la columna, en este caso el 1 y el 6.

B. Código

Como ya se ha dicho anteriormente, el backtracking es una técnica bastante común a la hora de resolver Sudokus. Estos algoritmos están ampliamente implementados tanto en Python como en lenguajes alternativos como Ciao Prolog.

Sin embargo, nuestro enfoque al basarse en estrtategias humanas, debido a la conveniencia hemos decidido implementar el algoritmo en Python.

Para evitar alargar el documento hemos decidido proporcionar un link al proyecto en github: SudokuSolver

VI. Conclusiones y Líneas Futuras

Desarrollando este algoritmo de reglas hemos observado que representa muy bien las estrategias que usamos los seres humanos por lo tanto es muy sencillo entender cómo funciona y saber por qué toma las decisiones que toma. Desafortunadaente, como todos los algoritmos basados en reglas simplemente aplica el conocimiento que se le otorga, no aprende ni puede desarrollar nuevas estrategias como lo hace por ejemplo el aprendizaje por refuerzo.

Sería interesante que el sistema pudiera aprender las estratagias humanas, en lugar de la selección manual por parte del usuario. Esto sería un paso más hacia el objetivo del trabajo, que sería obtener una inteligencia Artificial Explicable.

Al aumentar la dificultad del sudoku la cantidad de reglas aumenta también, además como tiene que deducir qué número va en la celda actual, utiliza razonamiento deductivo.

Este algoritmo no es capaz de resolver sudokus de nivel experto o superior, para ello necesitaría más reglas, mínimo tener 13 (depende de las reglas) de las más comunes [1].

Comparándolo con los algoritmos basados en restricciones, este algoritmo es menos eficiente y más complejo de implementar, pero más fácil de etender y comprender por qué se rellenan las casillas con ciertos números. Los basados en restricciones sí que pueden utilizar estrategias no conocidas ya que reciben el modelo del sistema y sus límites (restricciones) por ende sí que "aprende" nuevas estrategias.

References

[1] D. L. D. Scala, "How to solve a sudoku - a logical analysis and algorithmic implementation of strategically solving sudoku puzzles," M.S. thesis, Utrecht University, 2020. [Online]. Available: https://studenttheses.uu.nl/handle/20.500.12932/35782.

[2] *Sudoku.com*, https://sudoku.com/.

[3] G. McGuire, B. Tugemann, and G. Civario, *There is no 16-clue sudoku: Solving the sudoku minimum number of clues problem*, 2013. arXiv: 1201.0749 [cs.DS].

[4] M. Molina, *Models of reasoning*, Course of the Master Degree in Artificial Intelligence (lecture slides), Department of Artificial Intelligence, Universidad Politécnica de Madrid, 2023.

[5] P. Thagard and C. Shelley, "Abductive reasoning: Logic, visual thinking, and coherence," in *Logic and Scientific Methods: Volume One of the Tenth International Congress of Logic, Methodology and Philosophy of Science, Florence, August 1995*, M. L. Dalla Chiara, K. Doets, D. Mundici, and J. van Benthem, Eds. Dordrecht: Springer Netherlands, 1997, pp. 413–427. DOI: 10.1007/978-94-017-0487-8_22. [Online]. Available: https://doi.org/10.1007/978-94-017-0487-8_22.

[6] P. Johnson-Laird, "Deductive reasoning," *Annual Review of Psychology*, vol. 50, no. 1, pp. 109–135, 1999. DOI: 10.1146/annurev.psych.50.1.109.

[7] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. New Jersey: Prentice Hall, 2003.

[8] H. Simonis, "Sudoku as a constraint problem," 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:115950646.

[9] M. Schermerhorn, *A sudoku solver*, https://www.cs.rochester.edu/u/brown/242/assts/termprojs/Sudoku09.pdf.

[10] O. Eva, B. Desmond, and B. Dunka, "A hybrid backtracking and pencil and paper sudoku solver," *International Journal of Computer Applications*, vol. 181, no. 47, pp. 39–43, Apr. 2019. DOI: 10.5120/ijca2019918642. [Online]. Available: http://www.ijcaonline.org/archives/volume181/number47/30472-2019918642.