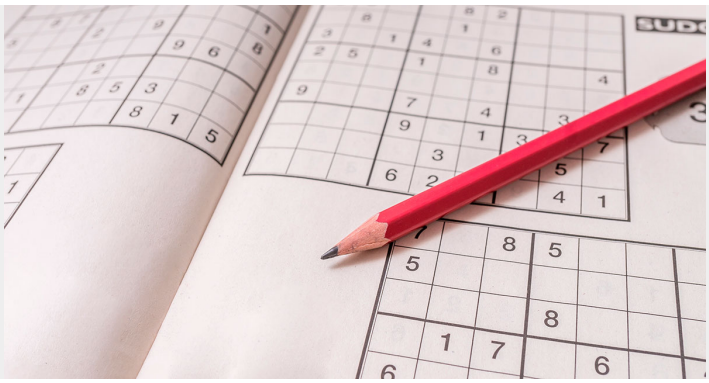
RESOLVIENDO SUDOKUS

Posted on 6 octubre, 2020 by Jesús Urias



Category: Ciencia

Cuentan que en Australia fue anulado un juicio sobre un asunto de drogas (una adicción a ciertos químicos) cuando las autoridades judiciales descubrieron que cinco (de doce) miembros del Jurado no podían dejar de jugar sudoku, olvidándose de enfocar su atención sobre los alegatos del proceso judicial en curso . Pareciera que se habían vuelto adictos al juego. ¿Es una droga el sudoku?

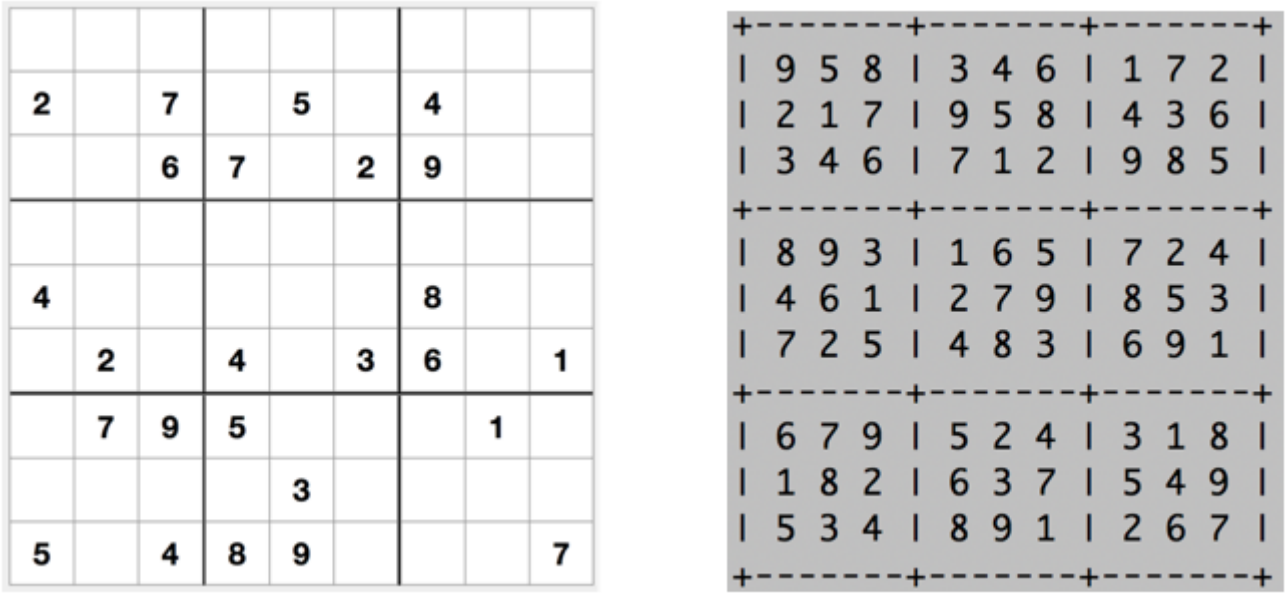
Como juego matemático el sudoku es un tipo particular del cuadrado latino del matemático Leonhard Euler (1707–1783) de Basilea. La versión moderna como un juego para lápiz apareció en 1979 como un diseño del arquitecto norteamericano Howard Garns. El juego se popularizó primero en Japón, en la década de 1980, donde le dieron el nombre de sudoku, que es la conjunción de las palabras sü = número y doku = solitario, resultando el neologismo japonés que designa al juego como un asunto de números solitarios. Esta y mucha más información sobre el sudoku puede consultarse en Wikipedia .

Quizás el éxito y proliferación por todo el mundo del sudoku radica en su amplio espectro de dificultades. Haciendo uso de solo un lápiz, el juego puede graduarse en una gama de niveles de dificultad que va desde los muy-muy fáciles hasta los prácticamente imposibles de resolver. Uno se

engancha con los fáciles, los va superando y al ir subiendo en dificultad, alcanzamos el nivel que nos hace producir la suficiente adrenalina como para desear ir en busca del sudoku que habíamos dejado a medias. Cuando esto nos pasa es porque ya somos adictos al juego.

¿De qué se trata el juego? Un sudoku se crea sobre un tablero cuadrado de 3×3 recuadros (o cajas),

de 3×3 casillas cada uno, constituyendo un tablero de $3^2 \times 3^2 = 81$ casillas. Un tablero de sudoku se muestra en la figura 1. Para propósitos del juego, en el tablero se identifican tres tipos de dominios. Un dominio es cada una de las nueve cajas 3×3 que se ilustran en el ejemplo de la figura 1. El segundo tipo de dominio es cada una de las nueve columnas de casillas y el tercero es cada uno de los nueve renglones de casillas.



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

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

Figura 1

En un sudoku cada casilla del tablero está iluminada de un color tomado de una paleta de nueve colores. Por comodidad los colores son designados por números enteros tomados de la paleta P = {1, 2, . . . , 9}. El tablero a la izquierda de la figura 1 solo tiene algunas pocas (25 de sus 81) casillas iluminadas. Las casillas que no han sido iluminadas en el tablero son las que se muestran sin una etiqueta numérica. El tablero iluminado en su totalidad a la derecha de la figura 1 es ejemplo de un sudoku.

No cualquier tablero completamente iluminado es un sudoku. Para serlo debe cumplir con la siguiente condición.

Todas las casillas de *cada uno* de los dominios (cajas, renglones y columnas) están iluminadas *sin repetir* color. (1)

Vemos pues que por la mera definición (1) el sudoku es un asunto de colores solitarios: un color no comparte su caja, ni su renglón, ni su columna con otro igual que él.

Una corrida de sudoku empieza al plantearse un problema que consiste en un tablero iluminado de manera incompleta. Un problema está bien planteado si tiene solución única. El tablero a la izquierda de la figura 1, que inicia con 25 casillas iluminadas, es ejemplo de un problema bien planteado. El reto para el jugador es iluminar las casillas sin color hasta llegar a completar el sudoku.

La meta es el tablero al lado derecho de la figura 1. Se sabe que resolver un sudoku "general" de $n^2 \times n^2$ casillas es equivalente al problema de "pégale a uno" (the hitting set problem) que es uno de los 21 problemas NP-completos de Richard Karp.

Por su complejidad, el sudoku fue elegido para poner a prueba el desempeño de un conglomerado de computadoras (a computer cluster). El matemático Gary McGuire y sus colaboradores de la Universidad de Dublín hicieron una búsqueda exhaustiva de problemas bien planteados que inician con solo 16 casillas iluminadas y en su búsqueda no encontraron ni uno (el descubrimiento fue confirmado en Taiwan al año siguiente). Desde entonces quedó bien establecido que el número mínimo de casillas iluminadas en problemas bien planteados es 17. Con el tiempo se ha integrado un catálogo que contiene un aproximado de 50,000 problemas mínimos (que son endiabladamente difíciles de resolver).

En las vacaciones de diciembre del 2019 encontré que resolver sudokus era una buena manera de sobrellevar las horas muertas (y frías). A partir de la propia definición del sudoku como colores solitarios (1) identifiqué las dos estrategias básicas para la solución de una clase muy amplia de problemas que yo llamo deterministas, pues a cada paso (correcto) que da el jugador, la ruta a la solución está determinada: se avanza con certeza en línea recta sin caer en incertidumbres que nos obliguen a consultar un oráculo.

> *El primer paso en la resolución de sudokus es darle mate a los deterministas*

El primer paso en la resolución de sudokus es darle mate a los deterministas. Con el fin de presentar las estrategias básicas y luego convertirlas en un algoritmo es necesario asignar un domicilio a cada una de las casillas en el tablero. Cada par de números $k = (x, y)$ $\{0,1,...,8\}^2$ es el domicilio de una casilla. La casilla superior izquierda del tablero es el origen, con domicilio (0, 0). Desde el origen, las y's avanzan hacia abajo y las x's avanzan hacia la derecha: es el orden lexicográfico latino: la x va de izquierda a derecha y la y va de arriba hacia abajo.

Para avanzar en la solución de un problema determinista, la estrategia consiste en identificar las restricciones impuestas por las casillas ya iluminadas sobre las casillas limpias. Importan los colores ya aplicados a las casillas de los tres dominios a los que pertenece una casilla k limpia. Designemos los colores ya aplicados de la siguiente manera. El conjunto de los colores ya aplicados a la caja de

k es Rk , Hk es la colección de colores ya aplicados al renglón de $_k$ y V_k los ya aplicados a la columna de k. El total de colores ya aplicados a los tres dominios de k, el conjunto R_k H_k V_k, decide los colores que es posible aplicar a la casilla k.

La condición de colores solitarios (1) que define a un sudoku, nos obliga a iluminar la casilla vacía k con alguno de los colores de la paleta reducida $Z_k = P - (R_k$ H_k $V_k)$, siendo P la paleta completa de nueve colores. Suponemos que el problema fue bien planteado y entonces la paleta reducida no puede estar vacía, $Z_k \neq 0$.

Solo existen dos casos de la paleta reducida Z_k para los cuales queda definido el color por aplicar a la casilla vacía k. Esos dos casos son los siguientes.

> Caso u: La paleta reducida Z_k solo tiene un color disponible para la casilla vacía k, lo cual se expresa como $card(Z_k) = 1$. Siendo opción única, se aplica ese color a la casilla k y deja de ser una casilla vacía.
>
> Caso v: La paleta reducida Z_k no es atómica, $card(Z_k) > 1$, pero entre los colores disponibles en Z_k hay un color que no aparece en la paleta reducida Z_k de cualquier otra casilla vacía k≠
>
> k en la caja ρ_k, o en ninguna otra k ≠ k vacía en el renglón α_k, o en ninguna otra k ≠ k vacía en la columna β_k. El único color de la paleta reducida Z_k que no repite como color posible en ninguna otra casilla limpia en alguna de sus propios dominios es un color solitario: cumple con la definición (1) y entonces se aplica el color a la casilla vacía k.

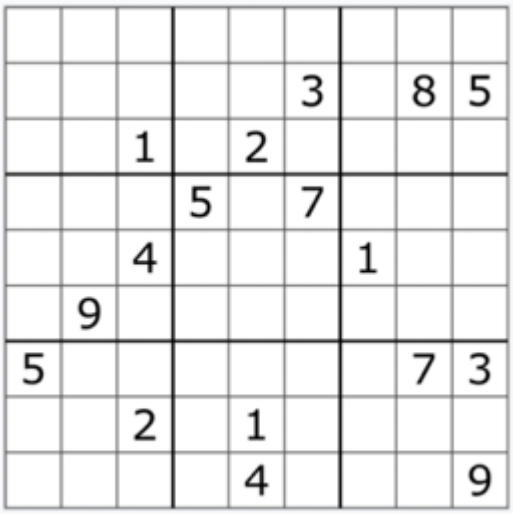
Con la práctica suficiente, un jugador llega a reconocer a ojo de pájaro las casillas limpias en el tablero que están en una de las dos condiciones anteriores. Un problema que se resuelve mediante la identificación reiterada sobre el tablero de las casillas limpias que están en condición u o en condición v pertenece a la clase de problemas deterministas. Los otros problemas son los diabólicos. Para resolver un diabólico hay que jugar adivinanzas para llegar a la solución.

Mediante el algoritmo listado en pseudo-código en la figura 2, se identifican a fuerza bruta (mediante recorridos exhaustivos del tablero) las casillas limpias que en el tablero están en situación u o v y así se resuelven —sin ningún contratiempo— todos los problemas deterministas. El problema mostrado a la izquierda de la figura 1 es determinista y la versión en Python del algoritmo en la figura 2 regresa en un tris la solución que se muestra a la derecha de la misma figura 1.

```
T = Tablero('problema.csv')
Cv = T.vacias()          # casillas vacias en el Tablero
while Cv.noVacio():
    for k in Cv:
        color = T.casou(k)
        if color.valido():
            T.paint(color)
            break
        color = T.casov(k)
        if color.valido():
            T.paint(color)
    Cv = T.vacias()
```

Figura 2

El segundo y último paso en la resolución de sudokus es complementar el algoritmo para deterministas con una estrategia para atacar a los diabólicos. En la figura 3 se muestra un problema mínimo (diabólico), con solo 17 casillas iluminadas, que fue diseñado como un reto especial para poner a prueba a los algoritmos basados en la fuerza bruta . A la derecha de la figura 3 se muestra el sudoku que es la solución al problema diabólico.



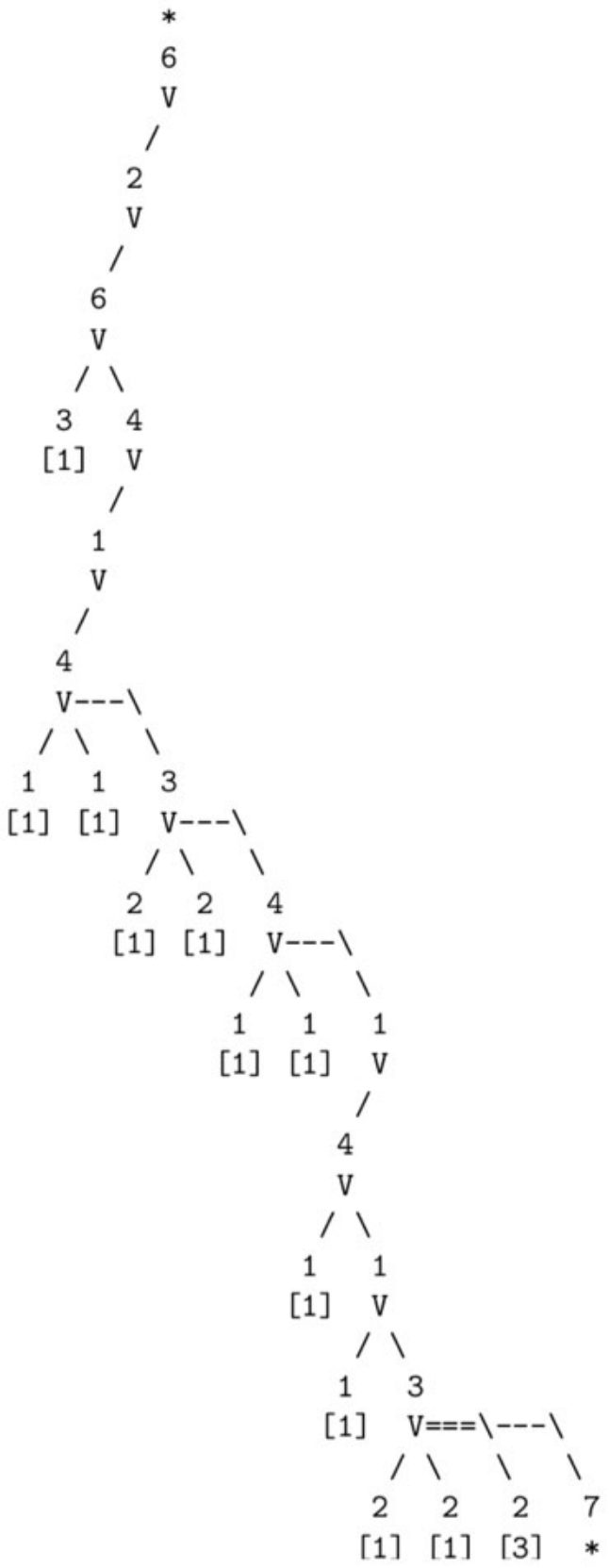
```
+-------+-------+-------+
| 9 8 7 | 6 5 4 | 3 2 1 |
| 2 4 6 | 1 7 3 | 9 8 5 |
| 3 5 1 | 9 2 8 | 7 4 6 |
+-------+-------+-------+
| 1 2 8 | 5 3 7 | 6 9 4 |
| 6 3 4 | 8 9 2 | 1 5 7 |
| 7 9 5 | 4 6 1 | 8 3 2 |
+-------+-------+-------+
| 5 1 9 | 2 8 6 | 4 7 3 |
| 4 7 2 | 3 1 9 | 5 6 8 |
| 8 6 3 | 7 4 5 | 2 1 9 |
+-------+-------+-------+
```

Figura 3

Nuestro primer algoritmo, tal cual se exhibe en la figura 2, es bueno para los deterministas. Cuando lo aplicamos a un problema diabólico como el de la figura 3, el proceso se atasca y ya no avanza hacia la solución. Al ir iluminando el tablero de un problema diabólico se llega a una configuración en la que la restricción de color solitario impuesta por las casillas ya iluminadas es insuficiente, y entonces no existen en todo el tablero casillas limpias que estén en la condición u o en la condición v. El tablero entra en una condición de indeterminación en la que no hay casillas limpias que tengan una opción única para el color solitario. Para salir del atolladero debemos consultar un oráculo que elija por nosotros alguno de los colores disponibles para alguna de las casillas limpias: un albur doble: casilla limpia y color solitario.

Esta situación hace que el proceso de resolución de un sudoku diabólico no siga una línea recta: en la condición de indeterminación cualquiera de las casillas limpias admite ser iluminada con más de un color sin generar conflicto —en la situación holgada en la que se encuentra el tablero— con las ya iluminadas. Una vez que el oráculo se decide por una de las casillas limpias, la elección de un color viable a ser solitarios define una ruta de exploración. Todas las rutas posibles en conjunto constituyen una estructura jerárquica simple (en forma de árbol) y lo que procede para encontrar la solución es recorrer el árbol de manera exhaustiva.

Un método concebido para ser exhaustivo es el recorrido en profundidad . Si una solución existe (que es única para un problema bien planteado), haciendo un recorrido en profundidad la vamos a encontrar. Si la solución al problema no es única, alguna vamos a encontrar.



La solución al problema diabólico en la figura 3 la obtuvimos siguiendo el recorrido en profundidad que se muestra en la figura 4. El “punto de salida” —la raíz del árbol— es el problema bien planteado, que se representa por el asterisco en la figura 4. Cada uno de los números en el diagrama (sin los corchetes) es el número de recorridos por el tablero en busca de casillas limpias en la condición u o en la v y que fueron exitosos. Partiendo del problema, el asterisco, se hicieron 6 recorridos exitosos y el siguiente recorrido (el séptimo) reporta que el tablero está en condición de indeterminación. Hay que pedir al oráculo que elija una casilla limpia y un color solitario. Esta acción corresponde a un vértice (un nodo) de ramificación del árbol, representado con la letra V en el diagrama. La decisión tomada en el punto V nos permitió hacer 2 recorridos exitosos y llegar a otra configuración de indeterminación. Aquí, la nueva decisión nos rindió 6 recorridos antes de llegar a una nueva indeterminación y la elección correspondiente nos permitió hacer 3 recorridos exitosos, para llegar, esta vez, a un callejón sin salida.

La última fue una mala elección, nos condujo a un configuración del tablero que tiene una casilla limpia con una paleta reducida vacía; una configuración del tablero en la que existe una casilla limpia que no admite un color solitario. Este callejón sin salida se indica en el árbol de la figura 4 por la anotación , que al no tener continuación es una hoja del árbol. Las otras configuraciones inadmisibles que pueden surgir son un color repetido en una caja , un color repetido en un renglón o un color repetido en una columna.

Cuando se nos presenta cualquiera de estas condiciones (caímos en una trampa), hay que re-

Figura 4

establecer la condición que el tablero tenía en el vértice previo V y poner a prueba otro de los colores disponibles como el color solitario y avanzar por otra rama del árbol. Continuamos con tales acciones hasta que eventualmente lleguemos a un tablero totalmente iluminado que nos da la solución al problema.

No es difícil elaborar un programa en Python que haga la búsqueda en profundidad que hemos descrito con ayuda del árbol en la figura 4. No hay misterios, está muy claro todo lo que hay por hacer. El algoritmo para resolver problemas diabólicos por búsqueda en profundidad se muestra en el pseudo-código en la figura 5. Los tiempos de solución para los problemas diabólicos con los que puse a prueba el programa son inapreciables a simple vista en mi PC. Así que no tuve la motivación para medir los tiempos de resolución. Para las pruebas busqué sin seguir un plan problemas mínimos de máxima dificultad.

```
sdk = Sudoku(problema)                    # Se carga el problema
ps = sdk.vacias                           # Numero de vacias
while ps > 0:                             # Sudoku incompleto
    error = sdk.pasoI()                   # u y v a todas las vacias
    error += sdk.repite()                 # color que repite es error
    if error:                             # rama incorrecta, regresar
        sdk, opciones, k = pila.pop()     # al vertice anterior
        sdk.celda[k] = opciones.pop()     # y elegir otra rama
        if opciones > 0:                  # Si aun quedan opciones
            pila.append([sdk,opciones,k]) # ahi puede estar la buena
        ps = sdk.vacias                   # las que quedan vacias
        continue                          # Avanzar por la nueva rama
    # ? No hubo avance y es necesario ramificar ?
    if sdk.vacias == ps:                  # sin avance --> adivinar
        k, n = sdk.menor()                # Primera vacia con menos opciones
        opciones, nT = sdk.albur(k)       # Albur para color en k
        pila.append([sdk,opciones,k])     # Por si hay que regresar
    ps = sdk.vacias                       # las que aun quedan vacias
sdk.out()                                 # solucion a stdout
```

Figura 5

Concluyo con la moraleja de que (a) no existe un problema bien planteado que escape al poder de un buen algoritmo (estoy hablando de los sudokus que tienen 9 × 9 casillas) y (b) disponer de este poder sobre los sudokus nos libera de la adicción. Esta última es una gran noticia. Así que por favor ayuden a difundir la buena nueva.

A los interesados en liberarse de la adicción y sentir el placer malsano de dar mate a todos los

sudokus les ofrezco una copia de mi código en Python. Solo escríbanme a juriash@gmail.com.

Referencias

Wikipedia, Sudoku. Url: https://en.wikipedia.org/wiki/Sudoku

Wikipedia, Set cover problem. Url: https://en.wikipedia.org/wiki/Set_cover_problem#Hitting_set_formulation

Gary McGuire, Bastian Tugemann and Gilles Civario. "There is no 16-clue sudoku: solving the sudoku minimum number of clues problem". (January 1st, 2012). Consultado en el url http://www.math.ie/McGuire_V1.pdf el 7 de enero de 2020

H.H. Lin and I-C. Wu. "No 16-clue sudoku puzzles by sudoku@vtaiwan_project" (September, 2013).

Wikipedia. Depth-first search. Url: https://en.wikipedia.org/wiki/Depth-first_search

Wikipedia, Sudoku solving algorithms. Url: https://en.wikipedia.org/wiki/Sudoku_solving_algorithms.