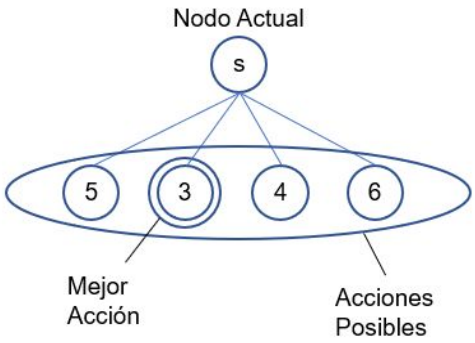
Solucionador de Sudoku usando BFS

Introducción

Se tiene como problemática el crear un algoritmo que sea capaz de resolver puzzles de sudoku a través de un grafo implícito de estados. Para este desafío, decidimos usar una variación del algoritmo BFS (Best-First Search) para recorrer los distintos nodos, tomando en cuenta ciertos parámetros para priorizar los nodos a recorrer en el siguiente nivel del árbol.

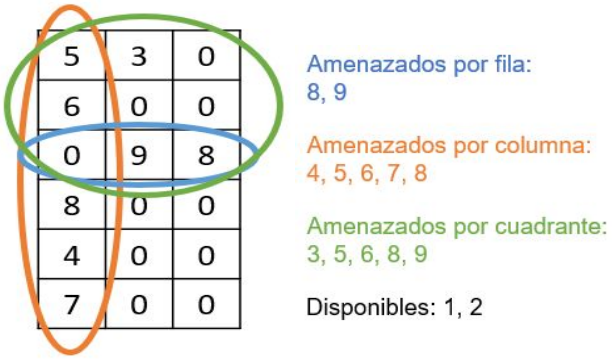
Desarrollo

El algoritmo escogido para resolver el sudoku es el BFS. La idea se centra en evaluar cada uno de los estados hijos posibles a partir del nodo actual en el que se encuentra el algoritmo, con tal de seleccionar un hijo en base a cierto criterio.

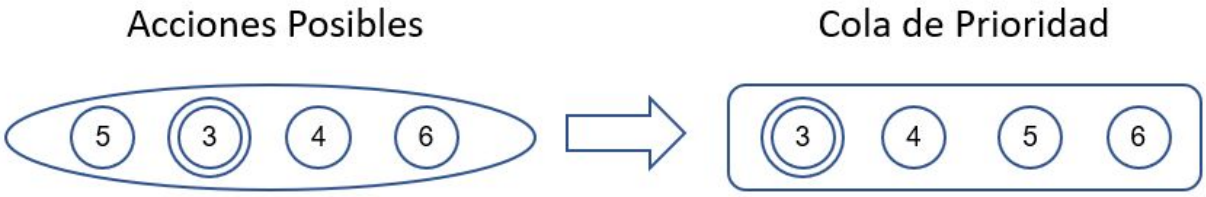


El criterio de selección de los mejores hijos de cada nodo se centra en el contador de espacios disponibles que tiene el tablero sudoku para poder insertar un número. En particular: “Mientras menor sea el contador, menor será el número de posibles jugadas dentro del cuadrante”, limitando el espacio de soluciones lo que acelera el hallazgo de una solución factible a dicho cuadrante.

Para determinar la prioridad de cada nodo hijo, se utilizó una aproximación de “amenazas” para cada casilla del sudoku, esto quiere decir que, en cada estado, se determinan las amenazas de cada casilla, las que representan los números ya asignados a la fila, columna y cuadrante de la casilla a consultar, lo que limita las opciones disponibles para rellenar dicha casilla. En base a esa disponibilidad, se determina el peso de cada nodo para evaluar su prioridad.



Para no perder el rastro de los otros nodos del mismo nivel y para no tener que correr nuevamente el algoritmo que analiza sus posibles movimientos, se hace una única llamada a la función que determina al mejor hijo, pero que ordena de en una cola con prioridad a todos los hijos para recorrer dicha lista a futuro (en caso de encontrar casos parcialmente correctos o derechamente fallidos).



A pesar de que aparenta ser eficiente, el algoritmo puede llegar a consumir bastante memoria en las observaciones realizadas durante las pruebas, por lo que se incluyó la librería “gc” para poder invocar manualmente el garbage collector de python al momento de eliminar nodos ya visitados y que se han confirmado como casos fallidos, para reducir el consumo de memoria del sistema que vaya a ejecutar el algoritmo.