SIA- TP1-

GRUPO7: Juan Pablo Rey, Santiago Camisay, Fernando Bejarano.

Contents

1 Problema asignado: CalcuDoku

El juego Calcudoku es una versión modificada del juego Sudoku. En el mismo se cuenta con un tablero de N por N casilleros. Dentro de este juego se tiene grupos, los cuales son conjuntos de casilleros junto con una operación matemática que, aplicada a dichos casilleros, debe dar un resultado definido por el mismo.

Además, también incluye las reglas propias del Sudoku original:

- los números con los que se puede completar el tablero son del 1 al N inclusive.
- no se puede repetir el mismo número por fila ni columna.

2 Llenado del tablero

Se llena todo el tablero de la siguiente manera: por cada fila se toman los números del 1 al N inclusive y luego se los mexcla en forma aleatoria. De esta manera se logra cumplir la restricción de que cada fila no tenga repetidos; el objetivo de esto es simplificar el problema y disminuir la cantidad de estados que se tienen que analizar por los distintos algoritmos. Al hacer esto último, el problema se reduce a intercambiar los valores de los casilleros dentro de una misma fila hasta que se logre cumplir las restricciones del juego.

3 Heuristicas (función h):

En las siguientes heurísticas, "G" representa la cantidad de grupos que no cumplen la restricción de permitir obtener el resultado al aplicar el operador de dicho grupo sobre los números de los casilleros de dicho grupo. Por otra parte, "C" representa la cantidad de columnas que no cumplen la restricción de no tener elementos repetidos. Debido a la manera en la que se resuelve el problema, se garantiza que por fila no haya repetidos y entonces no es necesario considerar que las filas tengan valores repetidos.

3.1 Admisibles

Las siguientes heuríticas fueron probadas con tableros de hasta 5 por 5 casilleros.

3.1.1 H1

La heurística consiste en la siguiente ecuación:

$$H1 = \left\lceil \frac{\max(G, C)}{2} \right\rceil$$

La idea detrás de este heurística es estimar la cantidad de intercambios entre casilleros considerando cual de los parámetros (G o C) presenta una mayor

cantidad de elementos incorrectos. Esto se obtiene mediante la función "máximo". Luego a este valor máximo, se lo divide por dos debido a lo siguiente:

- Caso de que el valor máximo sea C: en el caso la cantidad de columnas incorrectas se da en cantidadades pares luego las misma cantidad de intercambios que soluciona a una columna también soluciona a otra. Para los casos en los que esta cantidad no sea múltipla de dos, si se la divide por dos y se le calcula la función techo, se obtiene una cantidad menor o igual a la cantidad de pasos necesarios para solucionar el tablero
- En el caso en el que el valor máximo sea G: dado que se debe subestimar la cantidad de intercambios que faltan, se estima este valor considerando que con un intercambio se solucionan dos grupos a la vez.

3.1.2 H2

La heurística consiste en la siguiente fórmula:

$$H2 = \left\lceil \frac{C+G}{2} * \frac{1}{2} \right\rceil = \left\lceil \frac{C+G}{4} \right\rceil$$

Esta fórmula se obtiene calculando el promedio entre G y C; y luego dividiendo este resultado por dos. Al dividir este promedio, se obtiene una cantidad menor o igual a la cantidad real de pasos que faltan para obtener la solución.

3.2 No admisibles

Para el caso de tableros de 6 por 6 casilleros o superiores, no se logro llegar a la solución del tablero con las heurísticas anteriores debido a las restricciones de hardware y la capacidad de procesamiento disponibles en las computadoras utilizadas por el equipo durante la etapa de desarrollo. Debido a esto se decidió emplear la siguiente heurística para solucionar este inconveniente.

3.2.1 H8.

4 Estructuras de datos implementados en el problema

4.0.2 Tablero (Board.java)

Contiene los grupos, los valores de los casilleros y el parámetro N. Los valores están implementados en un BitSet para poder manipular los datos lo más rápido posible.

4.1 Estructura de un estado (CalcudokuState)

- la matriz de bits que representa el tablero.
- lista que contiene la representación en bits(almacenada en bitset) de cada numero que se puede emplear.
- lista de grupos. Las misma es final y no se copia cuando se copian estados (se comparte, no tiene sentido copiarlos por deep copy):

4.2 Estructura de un grupo

- result
- operador (MINUS,PLUS,MULTIPLY,DIVIDE,IDENTITY); "Identity" es simplemente el valor resultado.Este valor es un enum
- Lista de posiciones (i,j) de casilleros que forman el grupo.

5 Reglas

Se swapea cualquiera con cualquiera.s Las reglas se crean una sola vez al comienzo.Se descartan reglas simetricas.

6 Parser

Para generar el tablero inicial, parsea un string y devuelve el estado inicial. El estado inicial tiene el bitset en cero todo.

6.1 Algoritmos de búsqueda

6.1.1 desinformadas (no usan la función heuristica).

```
*bfs
*dfs
```

6.1.2 informadas (usan la funcion heuristica).

```
* A*.[listo.]
* iterativo (IDDFS): en gpsengine.getComparator hacer lo mismo que en dfs. Luego en GPSSol
```

6.2 Conclusiones

6.3 Pruebas de rendimiento

Para realizar las pruebas de rendimiento que se encuentran a continuación se tomaron 3 ejemplos de tableros para cada N, y se realizaron 5 corridas por cada

combinación de estrategia de búsqueda con cada una de las heurísticas (para el caso de IDDFS, BFS, y DFS no se emplea heurística según se vio en clase).

6.4 Bibliografía.

- http://www.conceptispuzzles.com/index.aspx?uri=puzzle/calcudoku/rules
- https://es.wikipedia.org/wiki/Kenken