Un puzzle un poco diferente

Daniel Antonio Quihuis Hernandez
Inteligencia Artificial

1 Representación del Estado

El estado se representa como una lista unidimensional de 16 elementos (1 al 16), donde cada posición corresponde a un lugar específico en la "esfera". Por ejemplo:

$$[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]$$

representa el estado objetivo. Las posiciones se asignan por filas: los primeros 4 elementos son la fila 1, los siguientes 4 la fila 2, etc.

2 Acciones Legales

En cualquier estado, las acciones son:

- **Rotar fila i (1-4) a la izquierda**: Desplazar los elementos de la fila i hacia la izquierda.
- **Rotar fila i (1-4) a la derecha**: Desplazar los elementos de la fila i hacia la derecha.
- **Rotar columna j (1-4) hacia arriba**: Desplazar los elementos de la columna j hacia arriba (como en una esfera).
- **Rotar columna j (1-4) hacia abajo**: Desplazar los elementos de la columna j hacia abajo.

Total: 16 acciones posibles (4 filas $\times$ 2 direcciones + 4 columnas $\times$ 2 direcciones).

3 Estado Sucesor

Dada una acción, el estado sucesor se calcula aplicando la rotación correspondiente:

- **Rotar fila i a la izquierda**:

$$[1, 2, 3, 4] \rightarrow [2, 3, 4, 1]$$

- **Rotar fila i a la derecha**:

$$[1, 2, 3, 4] \rightarrow [4, 1, 2, 3]$$

- **Rotar columna j hacia arriba**:

$$[1, 5, 9, 13] \rightarrow [5, 9, 13, 1]$$

- **Rotar columna j hacia abajo**:

$$[1, 5, 9, 13] \rightarrow [13, 1, 5, 9]$$

4 Costo Local

Cada acción tiene un costo de **1**, sin importar la fila/columna o dirección.

5 Cardinalidad del Espacio de Estados

El espacio de estados es $65,536$ (equivalente a 4^8).
Explicación:

- Cada fila puede rotarse en 4 estados (rotaciones 0, 1, 2, 3).
- Cada columna también puede rotarse en 4 estados.
- Total: $(4^4) \times (4^4) = 4^8 = 65,536$.

6 Admisibilidad de Heurísticas

- **Distancia de Manhattan**: **No es admisible**.
- **Número de piezas mal colocadas**: **Tampoco es admisible**.

7 Heurísticas Admisibles

Heurística 1 ($h_1(n)$):

$$h_1(n) = \frac{\text{Número de filas desalineadas} + \text{Número de columnas desalineadas}}{2}$$

Ejemplo: Si 2 filas y 3 columnas están desalineadas:

$$h_1(n) = \frac{2+3}{2} = 2.5$$

Heurística 2 ($h_2(n)$):

$$h_2(n) = \left\lceil \frac{\text{Piezas mal colocadas}}{4} \right\rceil$$

Ejemplo: Si 7 piezas están mal colocadas:

$$h_2(n) = \left\lceil \frac{7}{4} \right\rceil = 2$$

8 Demostración de Admisibilidad

Para $h_1(n)$:

- Cada acción corrige al menos una fila o una columna.
- $h_1(n)$ subestima el costo real, ya que se necesitan al menos $h_1(n)$ acciones para alinear filas y columnas.

Para $h_2(n)$:

- Una rotación puede corregir hasta 4 piezas.
- Si hay k piezas mal colocadas, se requieren al menos $\lceil k/4 \rceil$ acciones.
- Por lo tanto, $h_2(n) \leq h^*(n)$.

9 Dominancia entre Heurísticas

$h_1(n)$ **domina** a $h_2(n)$:

- Si una pieza está mal colocada, su fila y columna están desalineadas.
- $h_1(n)$ cuenta las desalineaciones de filas/columnas, mientras que $h_2(n)$ solo las piezas.
- Ejemplo: Si una fila entera está desalineada (4 piezas), $h_1(n) = 1$ y $h_2(n) = 1$.
- Pero si piezas están dispersas en múltiples filas/columnas, $h_1(n)$ crece más rápido que $h_2(n)$.

10 Búsqueda en Grafos vs Árboles

La búsqueda en grafos es superior:

- **Evita estados repetidos**: Las rotaciones pueden ciclar (ej: rotar una fila a la izquierda y luego a la derecha).
- **Reduce complejidad**: El espacio de estados es grande (4^8), y la búsqueda en grafos ahorra memoria al no revisitar nodos.
- **Eficiencia**: Encontrar la solución óptima requiere explorar menos nodos.