Sistemas de Inteligencia Artificial

Métodos de Búsqueda No Informados e Informados

Grupo 1

Integrantes:

Juan Pablo Civile
Esteban Ordano
Alvaro Crespo

Problema - Mahjong

Problema - Mahjong 2D

Descripción del juego

1) Cada turno, se saca un par de fichas del tablero.

Las fichas deben ser del mismo tipo y estar en cualquiera de los bordes izquierdo o derecho. No es necesario que estén en el mismo borde.

2) El juego se termina cuando no hay más movimientos posibles. Se gana cuando, además, no hay más fichas en el tablero.

Definición del Problema - Estados

- Estado inicial: Tablero con todas las fichas

- Acciones posibles y resultado: Para cada par de fichas del mismo tipo en los bordes del tablero, se puede sacar ese par de fichas para llegar a otro estado, con las mismas fichas en las mismas posiciones excepto por el par de fichas eliminadas.

Definición del Problema - Costo

Definimos el costo de ruta de un estado como la cantidad de fichas eliminadas.

El estado inicial tiene costo cero.

El costo de toda transición de un estado a otro es 2.

Análisis del problema

En el árbol de búsqueda, nodos en la misma altura tendrán el mismo costo de ruta.

Todas las soluciones están en la misma altura, y se conoce de antemano el valor del costo de ruta para llegar a ellas (cantidad de fichas del tablero).

Las soluciones son igualmente buenas, no hay soluciones mejores que otras.

Modelado e Implementación

- Estado: Array de double linked list
  - Cada elemento del array es una línea del tablero
- Tableros irresolubles: Existen tableros imposibles de resolver.
  - Ejemplo: sólo 4 fichas, de dos tipos, intercaladas (A-B-A-B)
- Hashing para guardar estados visitados

Heurísticas - 1) Trivial

- h = Cantidad de fichas en el tablero
- Es admisible
- Además, es acertada! (estima perfectamente el costo de llegar a la solución - lo cual es trivial)

Ya que sabemos cuál es "la mejor" heurística, las otras heurísticas que elegimos tienen como objetivo preferir expandir algunos estados antes que otros.

Heurísticas - 2) Ramas angostas

$$h(n) = \prod_{i=1}^{k} \binom{\#c_i}{2}$$

Tomando:

- k como la cantidad de tipos de fichas
- c_i como el conjunto de fichas elegibles para ser removidas del tipo i

- El objetivo es favorecer los caminos con menos elecciones posibles y expandir esos nodos primero, esperando que no haya *deadlocks.*
- Para que sea admisible, se toma el mínimo entre este valor y la heurística trivial

Heurísticas - 3) No contar eliminables

$$h(n) = \#F - \#R$$

- #F es la cantidad de fichas restantes
- #R es la cantidad de fichas que pueden ser removidas en la siguiente jugada.

Análisis y Resultados

- El problema tiene la particularidad de que todas las soluciones tienen el mismo costo; haciendo inútil la idea de minimizar el costo.
- La búsqueda por DFS es una muy buena estrategia, debido a que analiza primero los nodos más profundos. Por este motivo también BFS es una mala elección para solucionar el problema.

Análisis y Resultados (II)

- La búsqueda de Iterative Deepening mostró resultados ligeramente mejores que el BFS, pero mucho peores que el DFS.
- La estrategia A*, al buscar minimizar el costo de la solución, no es una buena opción para este problema. Los tiempos de ejecución y nodos expandidos resultaron ser semejantes al de BFS, independientemente de la heurística utilizada.

Análisis y Resultados (III)

- La estrategia Greedy con la heurística 3 mostró ser el mejor en una muestra de 100 tableros. El uso de la heurística le permite salir más rápidamente de las ramas que no llevan a una solución, (*deadlocks*), que es el mayor problema del DFS.

Conclusiones

Algoritmos que buscan soluciones óptimas, como BFS o A* no son una buena elección para la solución de este problema.

DFS es adecuado debido a que hace backtracking sólo cuando lo necesita (cuando se encuentra con un camino que no lleva a la solución).

Al igual que el DFS, el Greedy no hace backtracking innecesario, y con una buena heurística evita ramas muertas.