Sistemas de Inteligencia Artificial

Trabajo Especial:

Métodos de búsqueda no informados e informados

Instituto Tecnológico de Buenos Aires

Buenos Aires, Argentina

2018

Integrantes:

- Alan Arturo Hernández Gutiérrez alhernandez@itba.edu.ar 60438
- Kevin Chidiac kchidiac@itba.edu.ar 60431
- Romain Thibaut Marie Latron rlatron@itba.edu.ar 60440
- Ramiro Hernán Olivera Fedi rolivera@itba.edu.ar 56498

Profesores:

- María Cristina Parpaglione
- Alan Pierri

Introducción

El presente informe tiene como objetivo describir el proceso de construcción y los resultados obtenidos de distintos métodos de búsqueda informados y no informados para el juego ***Rolling Cubes***.

Sobre Rolling Cubes

El juego Rolling Cubes consiste de un tablero cuadrado de 9 casilleros y 8 piezas. El objetivo es rotar todas las piezas negras haciéndolas girar sobre el espacio vacío, hasta que todos muestren la cara blanca.

Una implementación de este juego se encuentra siguiendo la URL
http://www.minijuegos.com/juego/rolling-cubes

Definiendo el problema

El primer paso para el desarrollo del trabajo fue definir el problema de manera precisa, para luego poder desarrollar un modelo que lo represente y finalmente poder utilizar los distintos métodos de búsqueda para aproximar sus soluciones.

- **Estado inicial:** Tablero con 8 piezas mostrando su cara negra, sin pieza en el casillero del medio.

- **Conjunto de posibles acciones:** Dado un estado se puede rotar cualquier pieza que cumpla con la condición de que comparta uno de sus lados con el casillero vacío.

- **Modelo de transición:** Mover una pieza sobre el casillero vacío genera un estado en el que la posición en la que se encontraba ahora está vacía, y la posición que estaba vacía ahora contiene una rotación de la pieza movida.

- **Condición de solución:** Un estado es solución si todas las piezas del tablero son blancas.

Representando el problema

Habiendo ya definido el problema, pasamos entonces a representarlo.

Estado: Para la representación de los estados se optó que cada uno tuviera el tablero, representado como una lista (*List*) de cubos. Cada cubo contiene como un *enum* el color de la cara visible de la pieza que representa. De esta manera entonces, el tablero indica la cara visible de una pieza y su posición. Además el estado contiene también el índice del casillero vacío, para reducir el orden de la complejidad de acceso al mismo, una operación que se

realiza bastante a menudo.

Cabe aclarar además que se decidió representar el casillero vacío como un cubo de color *EMPTY*. Los valores posibles para los colores de un cubo están entonces compuestos por *WHITE, BLACK, WLEFT, WRIGHT, WUP, WDOWN, EMPTY,* donde los valores que inician con *W* representan una cara rayada (Blanca y Negra) con el lado blanco (W de white) en uno de los 4 lados (Izquierda, derecha, arriba y abajo, respectivamente).

Una vez que se logró representar un estado arbitrario, se construyó el estado inicial, que consiste en el tablero lleno de cubos con color *BLACK*, a excepción del centro (O el índice 4 en la lista que representa el tablero), que consta de un cubo con el valor *EMPTY*.

Cada estado además contiene las reglas permitidas para su tablero. La forma de las reglas se explicará con mayor detalle en las próximas secciones.

Reglas: Se representan todos los movimientos posibles para las piezas adyacentes al casillero vacío (4 movimientos). Pero las rotaciones permitidas se restringen de acuerdo a la posición del mismo. Hay entonces como máximo 4 reglas por estado. Las reglas representadas son (con respecto al cubo *EMPTY*): Rotar el cubo de arriba, el de la izquierda, el de la derecha, y el de abajo, con los objetos *ClickUp, ClickLeft, ClickRight* y *ClickDown* representandolos respectivamente.

Transición: Al aplicar una regla a un nodo (estado), se obtiene un nuevo nodo con una copia del tablero con la regla aplicada. De esta manera, el modelo de transición, aunque dependiente de la regla, puede describirse como:

- Ubicar en la posición de la pieza afectada por la regla una pieza vacía
- Ubicar en la posición de la pieza vacía una rotación de la pieza afectada con la regla, de acuerdo a una tabla de rotaciones

Terminación: Se define como la condición de terminación a la siguiente premisa: *Siendo el estado uno válido, todos los cubos son de color blanco, salvo uno que es vacío.* Si la condición se cumple, hemos encontrado una solución al problema planteado.

Reducción a problemas alternativos

Para comenzar a abordar el problema de la definición de funciones de costos y de heurísticas, buscamos soluciones a problemas similares.

El ***N-Puzzle*** (Un juego de deslizamiento de piezas que presentan un determinado orden inicial dentro de una cajita cuadrada) era un buen candidato para la reducción del problema de ***Rolling Cubes***. Las restricciones impuestas sobre las piezas del juego, si bien similares, son mucho más constrictivas en el N-Puzzle, haciendo de éste un juego más sencillo de representar.

Si bien no se logró la reducción, las heurísticas analizadas para este juego sirvieron de inspiración para el nuestro.

Algoritmos no informados implementados

Se implementaron los algoritmos Depth First Search (DFS), Breadth First Search (BFS), e Iterative Deepening (ID).

Algoritmos informados implementados

Se implementaron dos algoritmos informados: Greedy Search y A* (A estrella). Para el uso de ambos algoritmos se definieron distintas funciones de heurísticas **h(n)** representando la estimación del costo de llegar del nodo ***n*** al nodo de terminación.

Además para el algoritmo A* se definió la función ***g(n)*** representando el costo de haber llegado del nodo inicial al nodo ***n***.

Función de costos: *g(n)*

Se decidió utilizar como ***función de costos*** una función que incrementará en una unidad con cada regla aplicada. Es decir, utilizando los conceptos del juego, la cantidad de movimientos.

La elección de esta función fue casi trivial, ya que la misma implementación del juego que evaluamos (URL mencionada en explicación del juego) muestra a los usuarios en su interfaz la cantidad de movimientos como función de costos.

Heurísticas: *h(n)*

Para este trabajo se implementaron 2 (dos) heurísticas distintas. Ambas asumen que el tablero es válido. Esto lo logramos a partir de la construcción de estados válidos definiendo las reglas permitidas en cada uno de ellos.

Heurística de cubos negros

Surge a través de la idea de representar al objetivo del juego como la minimización de la cantidad de piezas con cara negra en el tablero. Se define la heurística como: ***h(n)*** = *2* □ ∑ *cubos con cara completa negra* + ∑ *cubos con media cara negra*

La heurística es bastante básica y no brinda mucha información sobre el juego, por lo que no es muy efectiva para los métodos de búsqueda informados. Esta heurística es admisible dado que el número total de movimientos para ordenar las piezas correctamente es al menos el número de piezas que no están bien ubicadas. También cumple con la condición

de consistencia, es decir que en cada paso, la función de costo será por lo menos igual al paso anterior, y eso porqué la heurística solo considera el costo de cada cubo sin considerar el resto del juego ni su posición.

Heurística Mejorada

Si bien la última heurística es buena, demora demasiado tiempo y es por lo tanto muy ineficiente por ser demasiado aproximativa. Por eso decidimos crear una nueva que tenga en cuenta además la cantidad mínima de movimientos para alcanzar el estado objetivo. Se tuvo en cuenta un conjunto de casos especiales. Por ejemplo, si tenemos un cubo en un borde que no puede ser rotado a blanco en un solo paso, notamos que la cantidad mínima de movimientos era de 12 aplicaciones de reglas, y desplazandolo 4 veces. Repitiendo este mecanismo para cada cubo, y teniendo en cuenta el color y la posición de cada cubo, tomamos el valor máximo entre el score y este nuevo valor como aproximación. Esta heurística no es consistente, para volver al ejemplo del cubo en el borde, si el valor de la heurística es el valor 12 por la cantidad de pasos mínima en cuanto a este cubo, y luego lo movimos y pasamos al valor correspondiente a la suma del costo de cada cubo, de ahí el valor de la heurística puede bajar en más que 1 y pues la función de costo será menor que en el paso precedente.

Resultados

Todos los resultados que se muestran fueron ejecutados sobre la misma computadora con un procesador Intel i7 2.2Ghz y 8GB de RAM dedicados. Todas las pruebas se hicieron sin límites de tiempo con el objetivo de evaluar el desempeño completo de los distintos algoritmos. Los resultados obtenidos se muestran en la Tabla 1 del anexo.

Conclusiones

Pudimos extraer muchas conclusiones del desarrollo del trabajo práctico. Una de las más notables es que la representación del problema es muy importante para el ahorro de memoria. La explosión combinatoria de la cantidad de estados en el *game tree* fuerzan a la optimización de la representación con respecto al espacio. En particular, nuestro grupo pudo haber utilizado una representación con *versionado de cambios*, en lugar de copiar el tablero en cada estado.

Por otro lado, pudimos analizar que las diferentes heurísticas brindan mayor o menor información sobre el problema. Es muy relevante encontrar buenas heurísticas para poder encontrar una buena solución. En particular, el análisis de admisibilidad es esencial si queremos alcanzar una solución óptima.

En relación a los resultados, puede apreciarse fácilmente que los algoritmos informados superan con creces la eficiencia de los no informados. A* anda muy lento comparado con Greedy Search, pero eso se puede explicar porque encuentra una solución óptima, o sea el camino más corto hasta el objetivo.

Anexo

Algoritmo	Profundidad	Estados generados	Nodos frontera	Nodos expandidos	Tiempo
DFS	112564	642721	410	642411	~218 min
BFS	97416	1970942	10526	1981468	~360 min
ID	112564	18415965	50349	18415965	~850 min
Greedy H1	160356	166563	0	166563	~144 min
Greedy H2	13385	13637	0	13637	~2 min
A* H1	36	1354536	425696	928840	~450 min
A* H2	36	526354	154268	372086	~200 min

Tabla 1 - Resultados del procesamiento con los distintos métodos de búsqueda

Nota: La heurística H1 se corresponde con la de cubos negros y la H2 se corresponde con la de heurística mejorada.