UNIVERSIDAD NACIONAL DE COLOMBIA

Maze Solver

Kevin Arturo Amaya Osorio

Ana Paulina Castillo Velásquez

Cristian Camilo Barreto Bejarano

Juan Pablo Ortega Bermúdez

Introducción a la inteligencia artificial

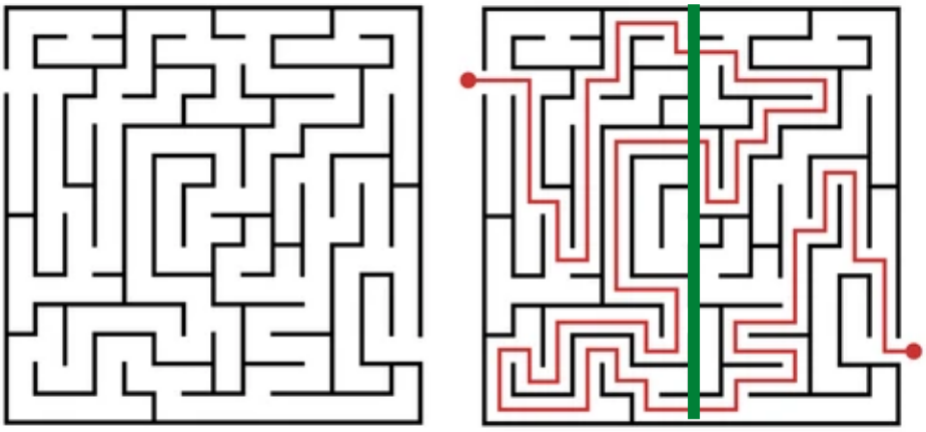
v

1. Explique qué es una heurística en problema del laberinto, proponga dos heurísticas y explique cómo pueden ayudar a resolver mejor el problema del laberinto.

   Los algoritmos heurísticos son aquellos que su objetivo es resolver tareas de manera más efectiva y rápida que algoritmos tradiciones, esto disminuyendo la optimización, la exactitud o la precisión. Enfocado en el problema del laberinto, una heurística es un algoritmo que genera camino aproximado de resolución sin que necesariamente sea este el más óptima. A continuación se mencionan dos algoritmo abordados en la página ***Less Wrong***

   - Cuello de botella

     El cuello de botella se basa en lo siguiente: Un subproblema que debe resolverse para resolver el problema principal. Particularmente para el laberinto lo podemos ver como una partición del espacio en dos secciones, una antes y una después de la linea. Para solucionar el problema es necesario atravesarla, por lo que nos encontramos con el nuevo objetivo de hallar una forma de llegar del punto de partida a la linea verde. Una vez definido el cuello de botella este método se puede combinar con otros algoritmos de búsqueda en una escala menor a la original.

     Este planteamiento puede ser utilizado dentro de los laberintos dado que reduce el tamaño de búsqueda al generar sub-objetivos, por lo que diferentes métodos de búsqueda pueden ser utilizados de manera conjunta en fracciones más pequeñas de espacio del problema general, enfocándose en los cuellos de botella más complicados.

   - Genético

     Los algoritmos genéticos describen un conjunto de técnicas inspiradas en la selección natural, como lo es: herencia, la mutación y el cruce. Se generan agentes con comportamientos erráticos y se premia a los agentes con mejores resultados permitiendo su cruce con otros agentes para la nueva generación.

Este algoritmo puede ser útil para el problema del laberinto ya que los agentes que exploren el mapa de manera eficaz serán los que ceden los genes para la siguiente generación, siendo esto una combinación de búsqueda en anchura y de profundidad ya que se está profundizando en cada generación pero se evalúan todos los posibles hijos escogiendo aquellos con mejores resultados, que particularmente en este problema se traduce en la cercanía con la meta.

2. Explique brevemente el algoritmo de búsqueda greedy. En particular, su estrategia de exploración y la representación de la frontera.

   El algoritmo greedy es un algoritmo de búsqueda donde un conjunto de recursos se divide recursivamente en función de la disponibilidad máxima e inmediata de ese recurso en cualquier etapa de ejecución. Se divide en dos grandes pasos: el análisis de los costos y optimización. Para el planteamiento del algoritmo es necesario definir una heurística; particularmente, para el problema de la resolución de laberintos, se toma como la distancia de Manhattan del punto n al nodo final, la cual nombraremos $f(n)$. El algoritmo escoge el siguiente paso según que su distancia con la meta sea la menor en comparación con las demás opciones.

   Representación de la frontera: La frontera es representada por una pila organizada por la distancia mínima al final.

   La estrategia de exploración es la siguiente

   a) Expandir el nodo actual.
   b) Calcular $f(n)$ para cada uno de los nodos hijos.
   c) Añadir a la pila el nodo con menor valor $f(n)$.
   d) Marcar al último nodo de la pila cómo el nodo actual n.
   e) Repetir pasos 1 al 3 hasta que el nodo objetivo sea encontrado o el nodo actual no tenga nodos hijos; en este caso donde el nodo actual no tenga hijos, se elimina de la cola el nodo actual y se repiten los pasos ignorando el nodo sin hijos.
   f) Usar la lista de nodos visitados para formar un camino del nodo objetivo al nodo de inicio.
   g) Retornar esta lista.

3. Explique brevemente el algoritmo A*. En particular, la estrategia de búsqueda y la representación de la frontera.

   A* o "A-star search" es un algoritmo de búsqueda informada, es decir qué usa información adicional a la proveída en la formulación del problema usando la siguiente heurística:

   $$f(n) = g(n) + h(n)$$

   donde

   - $h(n)$ = El costo estimado del nodo n hasta la meta
   - $g(n)$ = El costo de ir del nodo inicial al nodo n
   - $f(n)$ = Costo estimado del mejor camino desde el nodo n hasta el objetivo

   Generalmente la función $h(n)$ es un función qué calcula la distancia aproximada entre nodos, cómo por ejemplo la distancia de Manhattan o la distancia euclidiana.

   Representación de la frontera: La frontera es representada por una cola de prioridades de los nodos expandidos organizados de menor a mayor por $f(n)$.

   La estrategia de búsqueda es la siguiente [1]

   1. Expandir el nodo actual.
   2. Calcular la heurística $h(n)$ y el costo total $f(n)$ para cada uno de los nodos en la frontera.
   3. Añadir a la cola de prioridades toda los nodos en la frontera usando cómo prioridad a $f(n)$ ordenada de valores mínimos a máximos.
   4. Marcar al nodo actual cómo abierto/visitado y asegurarse de qué estos nodos no sean revisitados usando una lista.
   5. Guardar el nodo actual en una lista de nodos visitados. Esta lista es usada para formar un camino del nodo inicial al nodo objetivo.

6. Marcar al primer nodo de la cola de prioridades cómo el nodo actual n.
7. Repetir pasos 2 al 7 hasta que el nodo objetivo sea encontrado.
8. Usar la lista de nodos visitados para formar un camino del nodo objetivo al nodo de inicio.
9. Retornar esta lista.

Desempeño del algoritmo:

Completo: A* es completo siempre y cuando los nodos sean finitos y la heurística sea admisible.

Óptimo: A* es óptimo siempre y cuando la heurística sea admisible y monótona. Es decir la función nunca sobreestima el costo de alcanzar la meta, y a su vez sólo incrementa o decrementa. [2]

Complejidad de tiempo: Depende de la heurística aplicada al problema. Sin embargo dado a qué A* es una versión guiada de "Breadth- First algorithm" bajo una heurística admisible, este algoritmo tiene un peor caso escenario de $O(b^d)$

Complejidad de espacio: Por similar razonamiento , A* y "Breadth- First" comparten similar complejidad de espacio $O(b^d)$.

4. Un algoritmo de búsqueda se dice admisible si tiene garantía de retornar una solución óptima. Si la función heurística utilizada por A* es admisible, entonces A* es admisible. Explique que es una heurística admisible, y pruebe la afirmación previa.

   Una heurística admisible es aquella qué nunca sobreestima el costo de alcanzar la meta. En el caso de A*, significa que el camino óptimo del nodo n al objetivo nunca será menor que el valor aproximado por la heurística.

   La función heurística utilizada por A* es admisible si y solo si A* es admisible.

   Prueba:

   $\Rightarrow$

   Por contradicción, sea la heurística f de A* admisible y A* no admisible. Ya que A* no es admisible, entonces se tiene que hay un camino $C*$ dado por A* tal que $C* > C$ donde C es el camino óptimo. Ambos caminos divergen en un punto $n \in C$, tal que $f(n)$ es el costo optimo pero A* no lo expande, esta función f no sobreestima el costo ya que es admisible. Sin embargo esto es una contradicción ya que A* siempre expande el f con menor valor entonces no es posible que produzca el camino $C*$ ya que esta expandería n, lo cual genera una contradicción.

   $\Leftarrow$

   Por contradicción, sea A* admisible pero con una heurística f no admisible. Entonces se tiene que el camino C que produce A* es el óptimo, pero, como la heurística no es admisible, entonces el costo de algún nodo $n \in C$ tiene que ser mayor que su costo real, por lo tanto C no puede ser optimo, lo cual genera una contradicción.

5. Un algoritmo A es óptimamente eficiente con respecto a un conjunto de algoritmos alternativos Alts en un conjunto de problemas P si para cada problema P en P y cada algoritmo A en Alts, el conjunto de nodos expandidos por A al resolver P es un subconjunto (posiblemente iguales) del conjunto de nodos expandidos por A al resolver P. El estudio definitivo de la optimalizad eficiente de A* se debe a Rina Dechter y Judea Pearl [3]. Quienes consideraron una variedad de definiciones de Alts y P en combinación con la heurística de A* siendo meramente admisible o consistente y admisible. El resultado positivo más interesante que demostraron es que A*, con una heurística consistente, es óptimamente eficiente con respecto a todos los algoritmos de búsqueda similares a A* admisibles en todos los problemas de búsqueda no patológicos. Este resultado no se cumple si la heurística de A* es admisible pero no consistente. Explique en que consiste una heurística consistente, muestre un ejemplo de una heurística admisible pero no consistente, y pruebe el resultado de Dechter y Pearl [3].

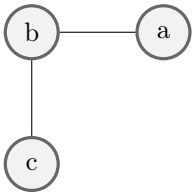
   Solución:
   (a) Sea $h(n)$ una heurística de un algoritmo A en un grafo G con nodo de inicio s y un nodo objetivo t. Entonces, h se dice **consistente** si para todo $n \in G$ y para todo hijo $n' \in G$ de n,

   $$h(n) \leq k(n, n') + h(n')$$

   donde K(n,n') representa el costo de llegar de n hasta n'. Una interpretación que se puede hacer a partir de la definición es que una heurística es consistente si no varía demasiado con respecto a al punto en el

que se encuentre.

(b) Sea G un grafo definido como,



a $\longrightarrow$ b $\longrightarrow$ c

Y sea $g(x, y)$ el costo del camino de x al nodo y definida como $g(a, b) = 5$, $g(b, c) = 2$, con c un nodo ogjetivo, y por consiguiente, $g(a, c) = 7$.
Así, sea $h(a) = 7$, $h(b) = 1$ y $h(c) = 0$. Entonces, como $h(a) = 7 = 7 = g(a, c)$ y $h(b) = 1 \leq 2 = g(b, c)$, h es una heurística admisible sobre (G, g), pero

$$h(a) = 7 > 6 = 5 + 1 = g(a, b) + h(b)$$

Es decir, h no es consistente sobre (G, g).

(c) Antes de demostrar la proposición, veamos que significa que una situación sea **no patológica**. una situación $I = (G, s, T, h)$ con G un grafo, s el nodo o estado inicial, T un conjunto de nodos objetivo, y, h una heurística, se dice **patológica** si siempre existe un camino donde para todo n en el camino, $h(n) \geq h*(n)$ donde $h*$ es el costo real desde el nodo n a un objetivo.

Veamos, ahora, que sea B un algoritmo que es consistente en cada situación, $I = (G, s, T, h)$ con G un grafo, s el nodo o estado inicial, T un conjunto de nodos objetivo, y, h una heurística consistente, entonces, en cualquier situación $I = (G, s, T, h)$ consistente y no patológica, todos los nodos expandidos por A* son también expandidos por B.
Sea una situación $I = (G, s, T, h)$ no patológica y consistente. Razonando por reducción al absurdo, supongamos que existe un nodo $n \in G$ que es expandido por A* y no por B. LUego, como I es no patológica, existe siempre un camino P en el cual $h(n') < h*(n')$ para todo n' nodo del camino P, entonces, siempre existe un camino P, tal que para todo n' nodo de P que no sea objetivo, $f(n') = g(s, n') + h(n') < g(s, n') + h(n') \leq C*$, con $C*$ el costo óptimo para llegar de s a t, es decir que todos los nodos que expande A*, r, cumplen que $f(r) < C*$, por la definición de A*.
Ahora, queremos crear un grafo G' nuevo con una nueva heurística h' tales que $I' = (G', s, T, h')$ es cosistente pero B no es consistente en I'. Definamos el grafo G' similar a G exceptuando que creamos una relación entre n y t con costo,

$$c := h(n) + \frac{1}{2}(C* - D)$$

con $D := mx\{f(m) : f(m) < C*\}$. Veamos que la nueva situación I' es consistente.
(i) h' es consistente: sean $k, m \in G$ cualesquiera. Si $k \neq t \neq m$ entonces la consistencia de h' se tiene por la consistencia de h. Supongamos que $k = t$. Entonces, $h(m) > g(m, t) + h(t) = g(m, t)$, pues de lo contrario,

$$g(m, t) > g(m, n) + c = g(m, n) + h(n) + \frac{1}{2}(C* - D)$$

es decir, $g(m, t) > g(m, n) + h(n)$. Pero h es consistente. contradicción. Por lo tanto, h' es consistente y I' es consistente.

Así, como, por la definición de máximo,

$$\begin{aligned} f(t) &= g(s, t) \\ &= g(s, n) + c \\ &= f(n) + \frac{1}{2}(C* - D) \\ &\leq \frac{1}{2}(C* + D) \\ &< \frac{1}{2}(C* + C*) \\ &= C* \end{aligned}$$

entonces, A* encuentra una solución en I' con costo $f(t) < C*$. Por lo tanto, como B es consistente en I, entonces, B es, en particular, admisible, y por consiguiente, B encuentra una solución óptima, es decir B encuentra una solución hasta t, con costo $C*$. pero como B no expande n, B no encuentra una solución óptima en I', es decir, B no es consistente en un instante I' consistente. Contradice que B es consistente sobre todo instante consistente.
Por lo tanto, B expande a n. □

Note la importancia de que I sea patológica, pues si I no es no patológica, puede existir un nodo n tal que $f(n) = C*$, lo que hace que $f(n)$ sea menor que D, esto evita que podamos hacer el mismo argumento para los n que cumplen $f(n) = C*$, restringiendonos a los n que cumplen $f(n) < C*$, haciendo que tengamos que considerar una optimalidad-eficiencia de A* con respecto a B sea más débil.

Referencias

[1] *Study on the Acceleration of Search Heuristics in Programmable Logic.*Federal University of Rio grande do sul institute of informatics computer engineering course. 2016.

[2] *Rudin - Principios de Análisis Matemático* Walter Rudin.

[3] *Generalized best-first search strategies and the optimality of A*.* Dechter, Rina; Judea Pearl (1985).