Sistemas de Inteligencia artificial
Métodos de búsqueda informados

Rodrigo Ramele, Juliana Gambini,
Juan Santos, Paula Oseroff,
Eugenia Piñeiro, Santiago Reyes

2022

Métodos de búsqueda informados

Introducción

Los métodos de búsqueda informados permiten guiar la búsqueda de una solución incorporando una estimación sobre la conveniencia o no de expandir un nodo dado.

- Para poder evaluar la conveniencia de expandir un nodo en vez de expandir otro/s se usa una función denominada **heurística**.
- Una heurística tiene como dominio el conjunto de estados y como imagen los reales
  $h : S \rightarrow \Re$.
- Dado un estado s, $h(s)$ es una estimación del costo que resta para alcanzar la solución.

Métodos de búsqueda informados

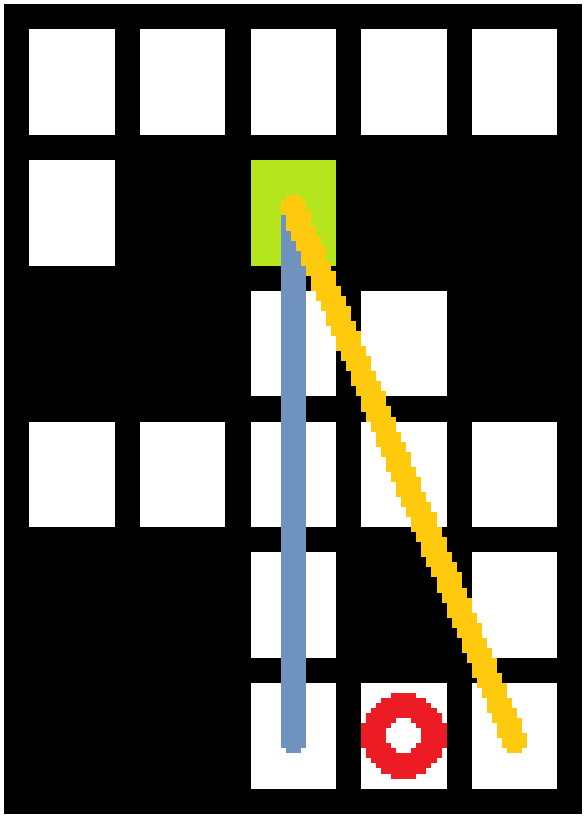
Heurística

Una función heurística debe satifacer:

- Si e es un estado objetivo
  $$h(e) = 0$$
- Si e no es un estado objetivo
  Si $\forall$ acción a el costo de realizar $a > 0$
  $$h(e) > 0$$
  Si $\forall$ acción a el costo de realizar $a \geq 0$
  $$h(e) \geq 0$$

Heurística

Veamos un ejemplo

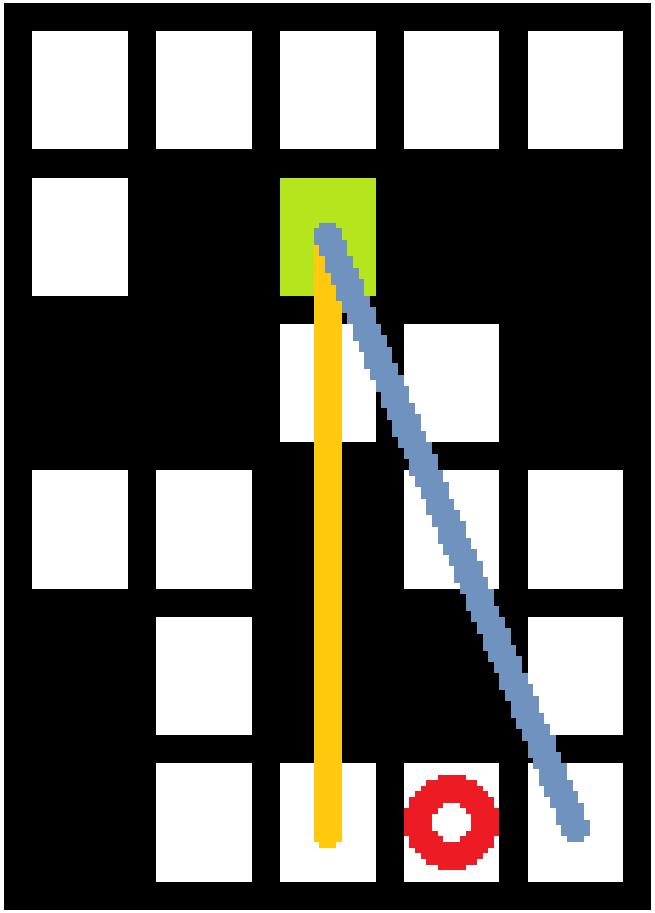


La distancia coloreada en ocre es mayor que la distancia coloreada en azul.
Por lo tanto, una heurística que tome en cuenta la distancia euclidea al objetivo guiará al agente a la celda 6,3 contra la acción que guía al agente a la celda 6,5 .

Luce como una buena heurística.

Heurística

Sin embargo ...



La misma heurística usada para el caso anterior no resulta adecuada en el contexto de este tablero .
Desde la celda 6,3 el costo de llegar al objetivo es mayor que el costo desde la celda 6,5

Ya no luce tan bien la heurística.

Búsqueda heurística local

Sea n_0 el nodo raíz de búsqueda y sea s el estado inicial que etiqueta a n_0.
Sea L_0 una lista vacía de nodos a la cual se le agrega el nodo n_0.

Algoritmo principal

1. LLamar a BúsquedaHeurísticaLocal(L_0)
2. Termina sin haber encontrado una solución.

Búsqueda heurística local

Sin retroceso

Algoritmo BúsquedaHeurísticaLocal(L)

1. Considerar al nodo n de L cuyo estado tenga el menor valor de heurística
2. Si el estado s de n es solución
   Fin de búsqueda con éxito.
3. Expandir n de acuerdo a las acciones posibles para el estado que lo etiqueta
4. Formar una lista de nodos $L_{sucesores}$ con los nodos obtenidos de la expansión de n.
5. LLamar a BúsquedaHeurísticaLocal($L_{sucesores}$)

Búsqueda heurística local

Con retroceso

Algoritmo BúsquedaHeurísticaLocal(L)

1. Mientras L no esté vacía
   Considerar al nodo n de L cuyo estado tenga el menor valor de heurística
   Si el estado s de n es solución
      Fin de búsqueda con éxito.
   Expandir n de acuerdo a las acciones posibles para el estado que lo etiqueta
   Formar una lista de nodos $L_{sucesores}$ con los nodos obtenidos de la expansión de n.
   LLamar a BúsquedaHeurísticaLocal($L_{sucesores}$)
   Remover n de L

Búsqueda heurística local

- No es óptima.
- Es completa si realiza retroceso (backtracking).
- Siempre es importante considerar la repetición de estados
- Una buena función heurística reduce significativamente su complejidad temporal y espacial.

Búsqueda heurística global

Sean **A** un árbol, **F** un conjunto de nodos frontera de **A** y **Ex** el conjunto de nodos explorados.

Búsqueda heurística global

Algoritmo BúsquedaHeurísticaGlobal

1. Crear **A**, **F** y **Ex** inicialmente vacíos.
2. Insertar el nodo raíz $\mathbf{n}_0$ en **A** y **F**.
3. Mientras **F** no esté vacía
   - Extraer el primer nodo **n** de **F**
   - Si **n** está etiquetado con un estado objetivo
     - Devolver la solución, formada por los arcos entre la raíz $\mathbf{n}_0$ y el nodo **n** en **A**
     - Termina Algoritmo.
   - Expandir el nodo **n**, guardando los sucesores en **F** y en **A**.
   - Reordenar **F** según el valor de la heurística del estado que etiqueta cada nodo.
4. Termina sin haber encontrado una solución.

Método A*

- La función de costo $g(n)$ permite obtener el costo del camino realizado desde el nodo raíz hasta el nodo actual n
- Notaremos como $f(n)$ a la función definida por la suma del costo de llegar a n y la heurística correspondiente al estado s que etiqueta a n:

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

donde $n.s$ denota el estado que etiqueta a n

Método A*

Algoritmo A*

1. Crear **A**, **F** y **Ex** inicialmente vacíos.
2. Insertar el nodo raíz $\mathbf{n}_0$ en **A** y **F**.
3. Mientras **F** no esté vacía
   - Extraer el primer nodo **n** de **F**
   - Si **n** está etiquetado con un estado objetivo
     - Devolver la solución, formada por los arcos entre la raíz $\mathbf{n}_0$ y el nodo **n** en **A**
     - Termina Algoritmo.
   - Expandir el nodo **n**, guardando los sucesores en **F** y en **A**.
   - Reordenar **F** según el valor de $f(n)$ para cada uno de sus nodos n.
4. Termina sin haber encontrado una solución.

Método A*

- $h()$ es **admisible** si $f(n)$ nunca sobreestima el costo real de la mejor solución que pasa por el nodo n.
- Dicho de otro modo, una heurística es **admisible** si
$$h(n.s) \leq h^*(n.s)$$
donde $h^*(n.s)$ es la heurística que corresponde a la solución óptima y
donde $n.s$ denota el estado que etiqueta a n.

¿Cuándo A* es óptimo?

- En cada estado debe haber un número finito de acciones posibles
- Todas las acciones tienen un costo mayor que 0.
- La heurísitica $h()$ debe ser admisible.

Método con costo y heurística pesados

Si consideramos la siguiente función $f()$

$$f(n) = (1 - w).g(n) + w.h(n.s)$$

el comportamiento del método de búsqueda cambiará de acuerdo al valor de w:

- $w = 0$ resulta en una búsqueda de menor costo primero
- $w = 1$ resulta en una búsqueda heurística global
- $w = 0,5$ resulta en una búsqueda A^*