Ayudantía 6

Heurísticas y Búsqueda con Adversario

IIC2613 - Inteligencia Artificial

ENTER

Contenidos

01

02

03

04

Heurísticas y Ejemplos

Admisibilidad y Consistencia

Algoritmo MiniMax

01
02
03
04

01.

Heurísticas

Heurísticas

- Enfoque utilizado para buscar soluciones a problemas tomando información o conocimiento previo.

Heurísticas

- Enfoque utilizado para buscar soluciones a problemas tomando información o conocimiento previo.
- No siempre encuentran la solución óptima, pero permiten llegar a una solución en un mejor tiempo. Se transa velocidad por optimalidad.

Heurísticas

- Enfoque utilizado para buscar soluciones a problemas tomando información o conocimiento previo.
- No siempre encuentran la solución óptima, pero permiten llegar a una solución en un mejor tiempo. Se transa velocidad por optimalidad.
- Buscan no perder tiempo en cosas que probablemente no ayuden a llegar a una solución. Especie de atajo o *short-cut* a la solución.

Heurísticas

- Enfoque utilizado para buscar soluciones a problemas tomando información o conocimiento previo.
- No siempre encuentran la solución óptima, pero permiten llegar a una solución en un mejor tiempo. Se transa velocidad por optimalidad.
- Buscan no perder tiempo en cosas que probablemente no ayuden a llegar a una solución. Especie de atajo o *short-cut* a la solución.
- Se busca **estimar** cual es la mejor decisión en cada paso de la búsqueda de una solución.

Heurísticas

- Enfoque utilizado para buscar soluciones a problemas tomando información o conocimiento previo.
- No siempre encuentran la solución óptima, pero permiten llegar a una solución en un mejor tiempo. Se transa velocidad por optimalidad.
- Buscan no perder tiempo en cosas que probablemente no ayuden a llegar a una solución. Especie de atajo o *short-cut* a la solución.
- Se busca **estimar** cual es la mejor decisión en cada paso de la búsqueda de una solución.
- La Función Heurística, estima que tan cerca está un estado *s* del estado final G.

Ejemplos Heurísticas

- Elegir el camino conocido a un lugar, en vez de tomar otro que es más rápido.

Ejemplos Heurísticas

- Elegir el camino conocido a un lugar, en vez de tomar otro que es más rápido.
- No comprar un producto del que escuchamos malas opiniones.

Ejemplos Heurísticas

- Elegir el camino conocido a un lugar, en vez de tomar otro que es más rápido.
- No comprar un producto del que escuchamos malas opiniones.
- Llevar paraguas cuando el día está nublado.

Ejemplos Heurísticas

- Elegir el camino conocido a un lugar, en vez de tomar otro que es más rápido.
- No comprar un producto del que escuchamos malas opiniones.
- Llevar paraguas cuando el día está nublado.
- Elegir un producto porque su envase se ve de mejor calidad.

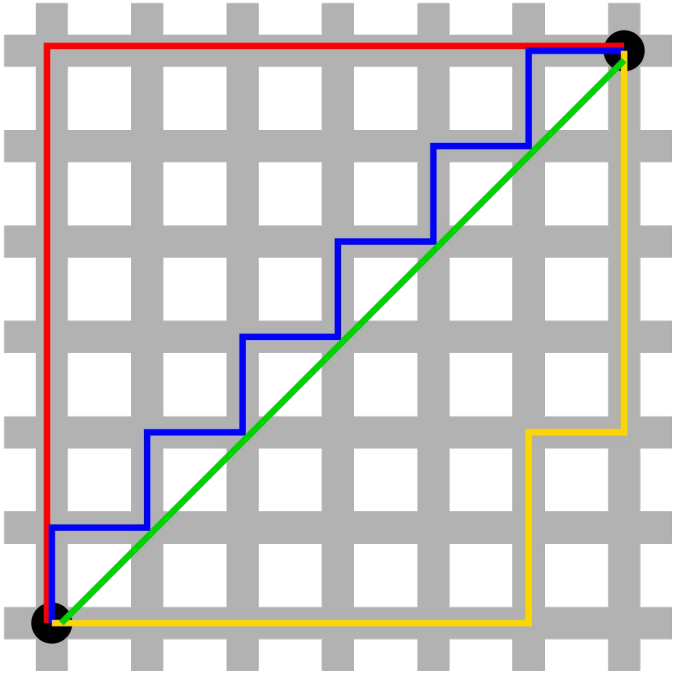
Ejemplos Heurísticas

- Elegir el camino conocido a un lugar, en vez de tomar otro que es más rápido.
- No comprar un producto del que escuchamos malas opiniones.
- Llevar paraguas cuando el día está nublado.
- Elegir un producto porque su envase se ve de mejor calidad.
- Distancia Euleriana

Ejemplos Heurísticas

- Elegir el camino conocido a un lugar, en vez de tomar otro que es más rápido.
- No comprar un producto del que escuchamos malas opiniones.
- Llevar paraguas cuando el día está nublado.
- Elegir un producto porque su envase se ve de mejor calidad.
- Distancia Euleriana
- Distancia de Manhattan

Distancia Manhattan



01

02

03

04

01
02
03
04

02.

Admisibilidad y Consistencia

Admisibilidad

La función heurística h es admisible si para todo estado s, se cumple que:

$$h(s) \leq h^*(s)$$

Con $h^*(s)$ el costo de un camino óptimo desde s a un estado objetivo

En otras Palabras

h es admisible si nunca sobre-estima, es decir si el costo estimado por la heurística no supera el costo real desde s hasta un objetivo

01

02

03

04

Teorema

Si h es admisible, entonces A^*, usado con h encuentra una solución óptima si esta existe

Consistencia

La función heurística h es consistente si y sólo si se cumple que:

- $h(s) = 0$, para todo $s \in G$

No se sobre-estima en los objetivos

- $h(s) \leq c(s, s') + h(s')$, para todo vecino s' de s

La variación del h de dos estados no es mayor que el costo de ir de uno al otro

01

02

03

04

Teorema

Si h es consistente, entonces h, es admisible

Ejercicio

Demuestre que la suma de distancias Manhattan es una heurística admisible para el problema del puzzle de 8.

01
02
03
04

03.

Algoritmo MiniMax

MiniMax Terminos Importantes

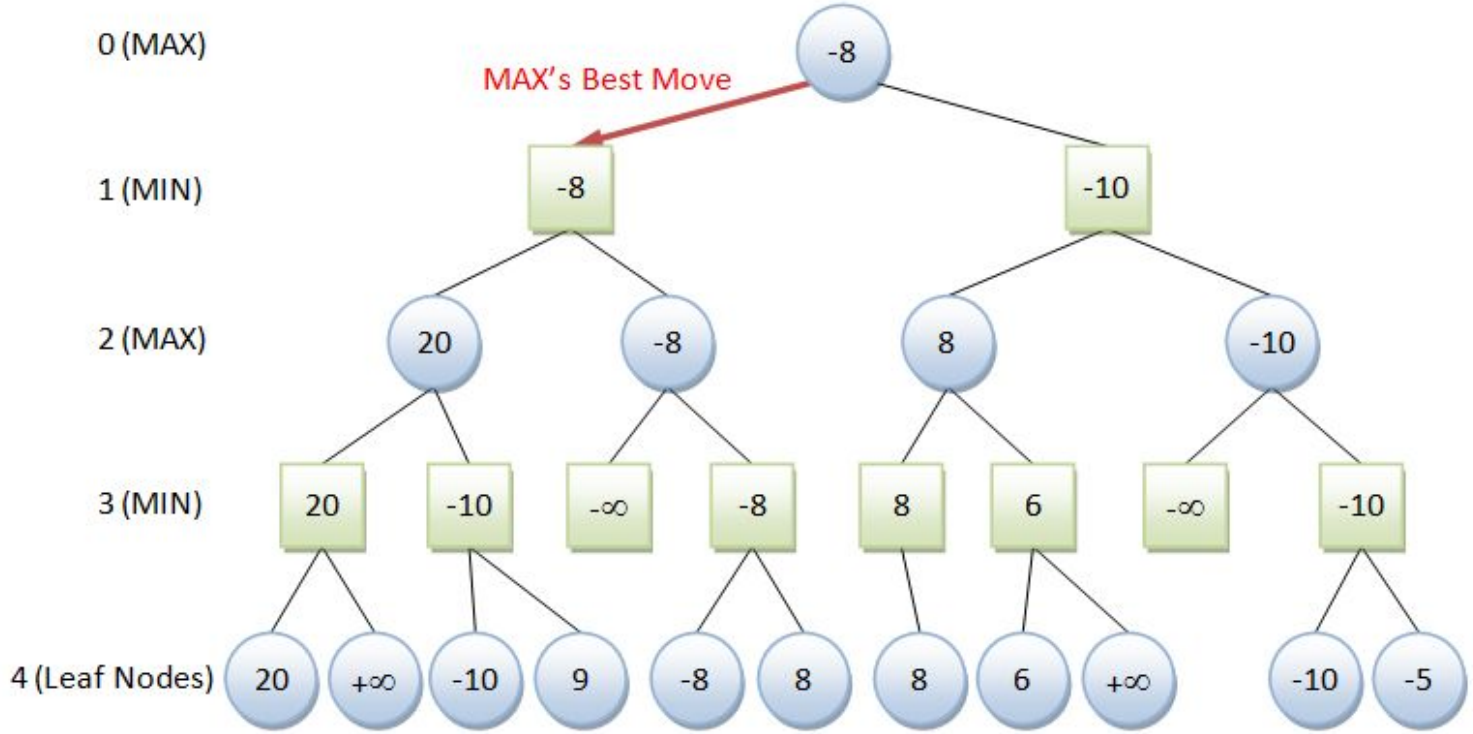
Maximizing Player: Trata de conseguir la mayor puntuación posible

Minimizing Player: Hace lo opuesto, trata de conseguir la menor puntuación posible

Cada estado del tablero tiene un valor asociado:

- Si el maximizador tiene ventaja, la puntuación del tablero tendrá que ser un valor positivo
- Si él minimizador tiene la ventaja en ese estado del tablero, tendrá un valor negativo
- Estos valores los determina una **función de evaluación**

MiniMax


0 (MAX)
MAX's Best Move
-8
1 (MIN)
-8
-10
2 (MAX)
20
-8
8
-10
3 (MIN)
20
-10
-∞
-8
8
6
-∞
-10
4 (Leaf Nodes)
20
+∞
-10
9
-8
8
8
6
+∞
-10
-5

Zero-sum Definición

Los juegos de suma cero son un caso especial de suma constante de juegos, en el que las elecciones de los jugadores no pueden aumentar ni disminuir los recursos disponibles.

En los juegos de suma cero, el beneficio total para todos los jugadores del juego, para cada combinación de estrategias, siempre suma cero (más informalmente, un jugador se beneficia solo a expensas del otro).

PseudoCódigo de MiniMax

1. Revisamos si el depth es 0 o si se acabo el juego
2. Si el jugador es el maximizing Player:
    a. Dejamos el value como -infinito.
    b. Recorremos todos los hijos por nodo y vemos cual es el puntaje del nuevo tablero
    c. Si el puntaje es mayor que value, lo cambiamos
3. Se hace lo mismo que el anterior pero si el puntaje es menor que value, lo cambiamos

PseudoCódigo de MiniMax

3. Si el jugador es el minimizing Player:
    a. Dejamos el value como infinito.
    b. Recorremos todos los hijos por nodo y vemos cual es el puntaje del nuevo tablero
    c. Si el puntaje es menor que value, lo cambiamos

PseudoCódigo de MiniMax

```
function Minimax (node, depth, maximizingPlayer) is
    if depth == 0 or node is terminal node then
        return state.score()
        # score o función de evaluación estima el valor de
        # un nodo y "predice" el valor

    if maximizingPlayer then
        value = -math.inf
        for each child of node do
            value = max(value, minimax(child, depth -1, FALSE))
        return value
    else: # minimizingPlayer
        value = math.inf
        for each child of node do
            value = min(value, minimax(child, depth -1, TRUE))
        return value
```

01

02

03

04

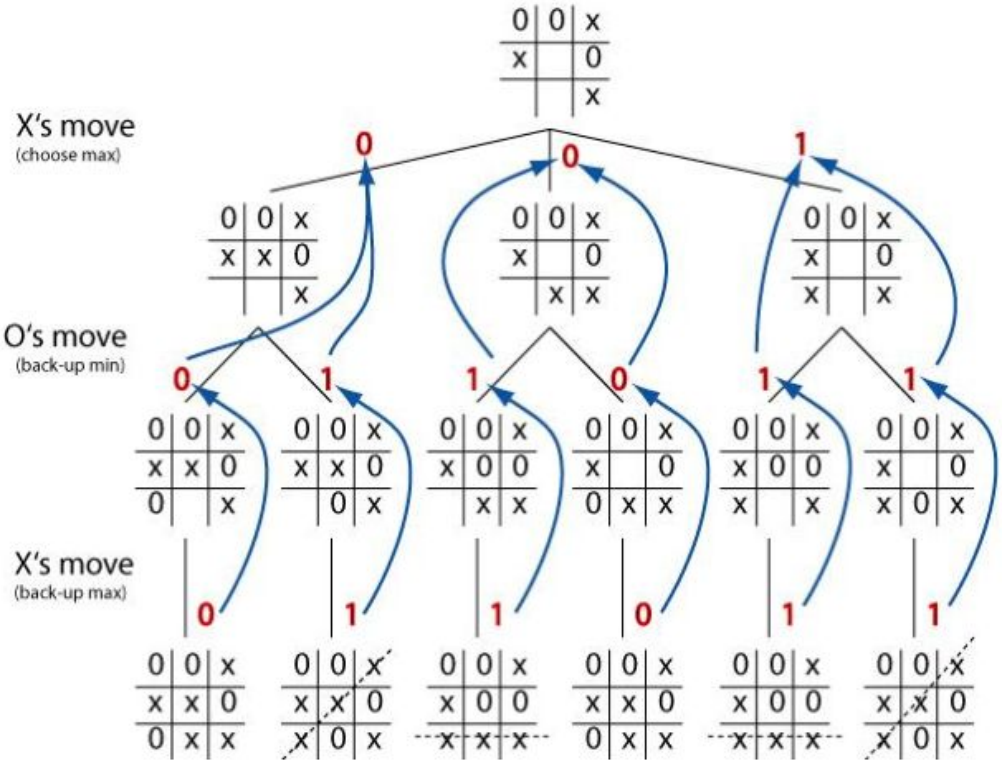
Poda alpha-beta

1. Revisamos si el depth es 0 o si se acabo el juego
2. Si el jugador es el maximizing Player:
    a. Dejamos el value como -infinito.
    b. Recorremos todos los hijos por nodo y vemos cual es el puntaje del nuevo tablero
    c. Si el puntaje es mayor que value, lo cambiamos
        i. Decimos que alpha es igual al maximo entre value y alpha
    d. Si alpha es mayor o igual a beta, hacemos break

Poda alpha-beta

3. Si el jugador es el minimizing Player:
    a. Dejamos el value como infinito.
    b. Recorremos todos los hijos por nodo y vemos cual es el puntaje del nuevo tablero
    c. Si el puntaje es menor que value, lo cambiamos
        i. Decimos que beta es igual al minimo entre value y beta
    d. Si alpha es mayor o igual a beta, hacemos break

Ejemplo con el juego del gato

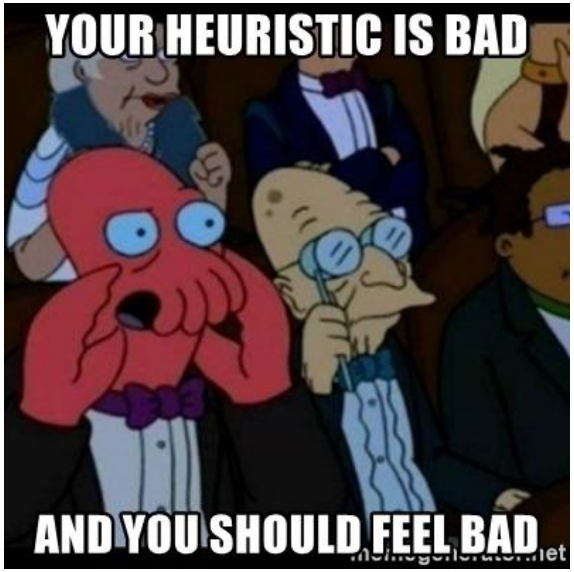


Score = +1 (gana IA)

Score = -1 (pierde IA)

Score = 0 (empate)

¡Muchas gracias!



MAX wants to maximize its utility
So, MIN calculating how to minimize MAX's utility