Razonamiento y Planificación Automática

César Augusto Guzmán Álvarez

Doctor en Inteligencia Artificial

Tema 7 : Búsqueda multiagente

Sesión 2/2

unir LA UNIVERSIDAD EN INTERNET

Universidad Internacional de La Rioja

Resumen – Tema anterior

Tema 7 : Búsqueda multiagente

X	O	**X**
O	O	**X**
O	X	**X**

Sesión 1 :

- Asunción de los problemas a resolver
- Búsqueda minimax
- Práctica Tres en Raya en Python

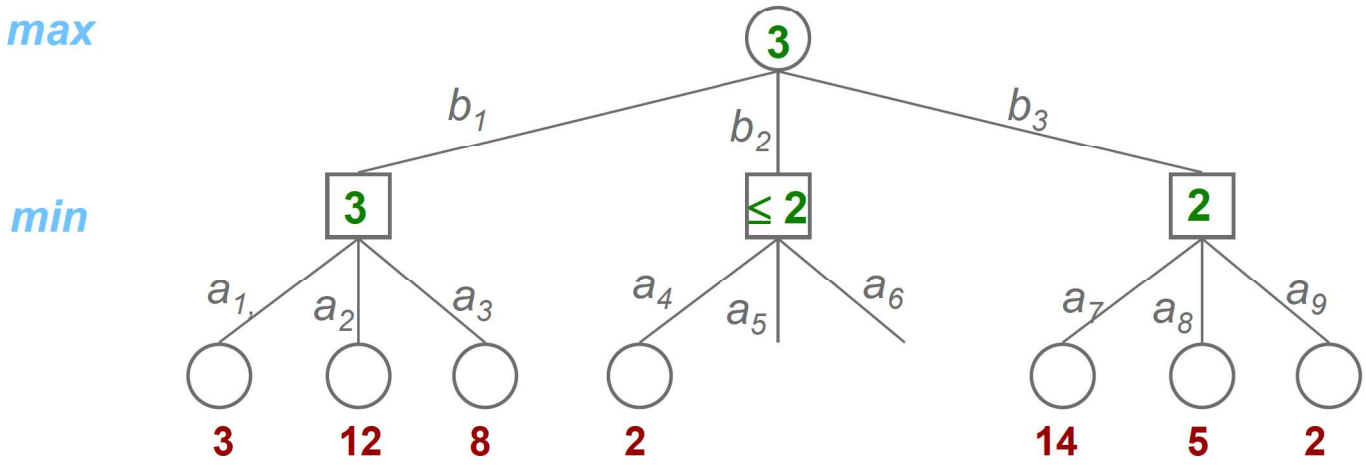
Sesión 2 :

- La poda alfa-beta
- Búsqueda expectiminimax
- Práctica Tres en Raya con poda alfa-beta

Índice

La poda alfa-beta

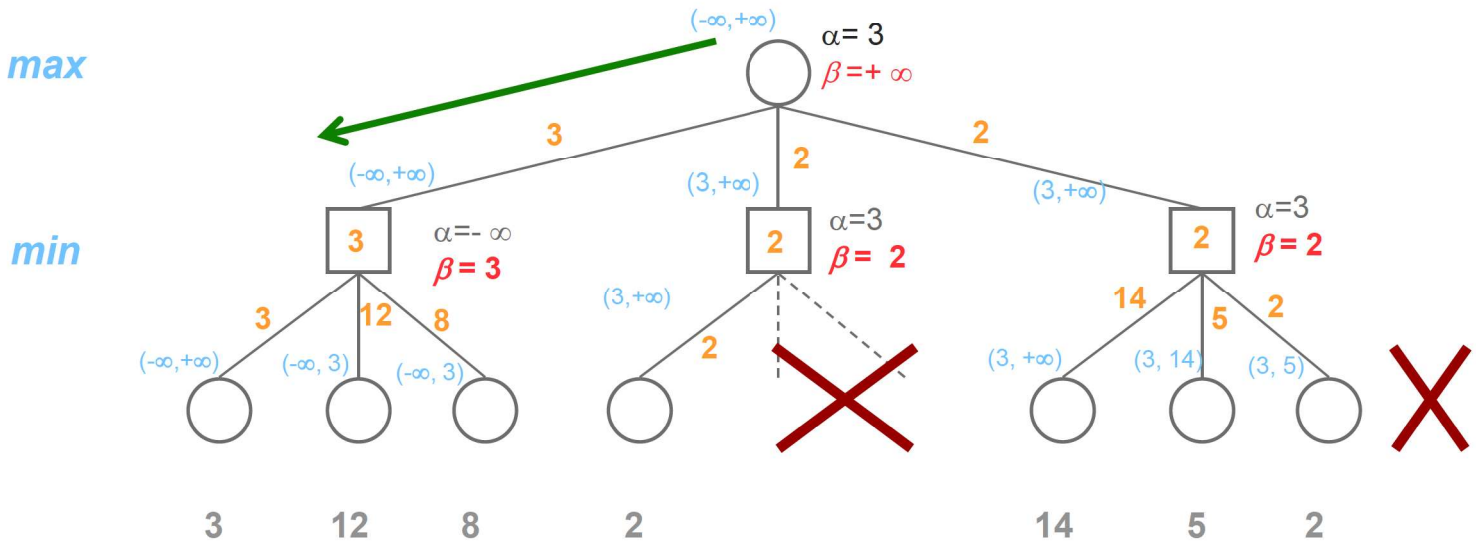
En Minimax el número de estados a explorar es **exponencial** al número de **movimientos**



- **Alfa** mejor valor que **max** puede garantizar en el nivel actual o superior.
- **Beta** mejor valor que **min** puede garantizar en el nivel actual o superior.

La poda alfa-beta - Ejemplo

- Simulación del ejemplo con el algoritmo :
  - **Negro** / **Rojo** : valores de α y β **dentro** de la función miniMax
  - **Azul**: valores de los **parámetros** α y β (entrada) a la función miniMax
  - **Amarillo**: valores **devueltos** por la función de utilidad miniMax



La poda alfa-beta - Algoritmo

```
alfaBeta(estado, [profundidad], esMax, alfa, beta):

1:   si esTerminal(estado) entonces
2:      retornar utilidad(estado)
3:   sucesores = expandir(estado)
4:   si esMax entonces
5:      mejorUtilidad = -INFINITO
6:      para cada s en sucesores :
7:         utilidad = alfaBeta(s, [profundidad+1], false, alfa, beta)
8:         mejorUtilidad = max( mejorUtilidad, utilidad)
9:         si mejorUtilidad >= beta entonces
10:           break
11:        alfa = max(alfa, mejorUtilidad)
12:     retornar mejorUtilidad
13:  sino
14:     mejorUtilidad = +INFINITO
15:     para cada s en sucesores :
16:        utilidad = alfaBeta(s, [profundidad+1], true, alfa, beta)
17:        mejorUtilidad = min( mejorUtilidad, utilidad)
18:        si mejorUtilidad <= alfa entonces
19:           break
20:        beta = min(beta, mejorUtilidad)
21:     retornar mejorUtilidad
```

La poda alfa-beta - Algoritmo

Análisis:

- Siempre produce el mismo resultado que sin la poda
- La eficiencia depende del orden en el que se exploran los nodos
- Complejidad (factor de ramificación b = 10; profundidad explorada d = 4)

  – Poda alfa-beta (peor caso) : $O(b^d)$: (10^4 = 10.000 nodos)
  (Igual que miniMax)

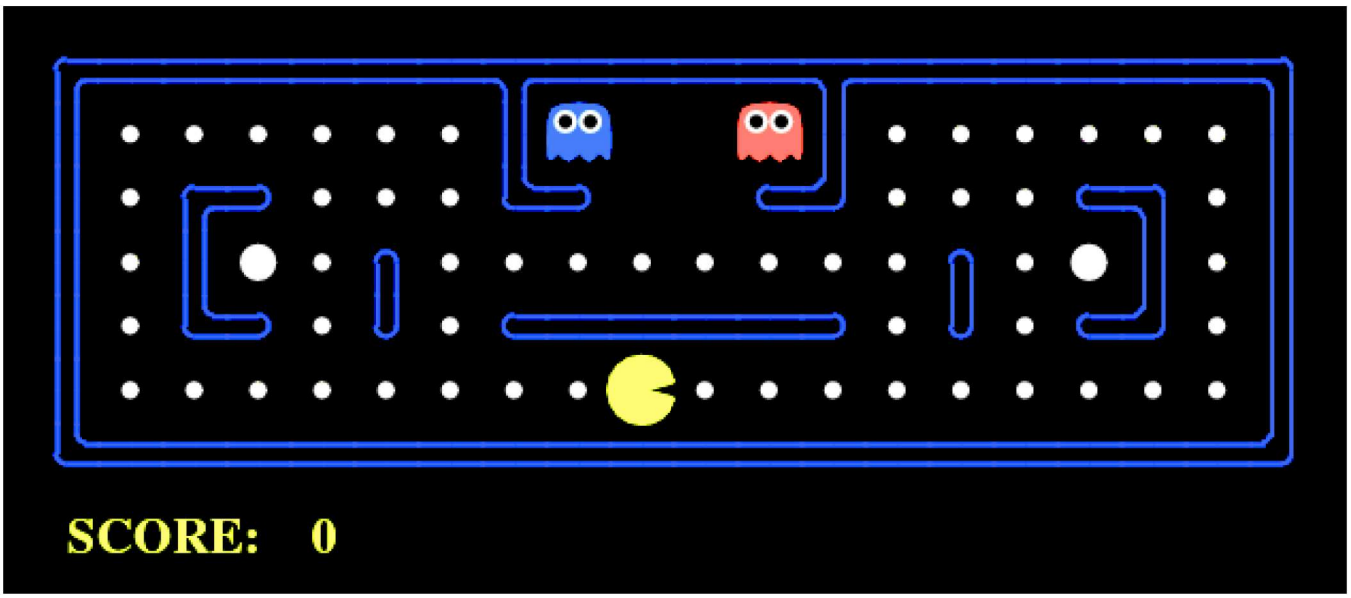
  – Poda alfa-beta (mejor caso): $O(b^{d/2})$: $O(\sqrt{b^d})$: (10^2 = 100 nodos)

  – Poda alfa-beta (caso medio): $O(b^{3d/4})$: (10^3 = 1.000 nodos)

Índice

Búsqueda expect minimax

STOCHASTIC GAMES



Pacman

Búsqueda expectiminimax

- Es una derivación del algoritmo minimax
- Aplicado en juegos de:
  - ✓ suma cero,
  - ✓ bipersonal,
  - ✓ información incompleta,
  - ✓ y no determinísticos
- Utiliza una función de evaluación

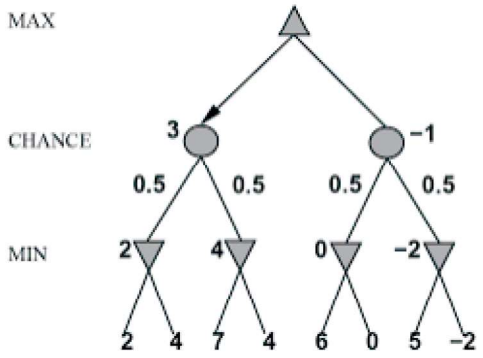


Ilustración 4: Ejemplo del algoritmo Expectiminimax

$$\text{ExpectiMinimax}(n) = \begin{cases} \text{si } n \text{ es nodo max} & max \text{ nodo n sucesor} \\ \text{si } n \text{ es nodo min} & min \text{ nodo n sucesor} \\ \text{si } n \text{ es nodo change} & promedio\ [\ P(n') * ExpectiMinimax(n')\ ]\ \forall n' \in sucesores(n) \end{cases}$$

Búsqueda expectiminimax - Ejemplo

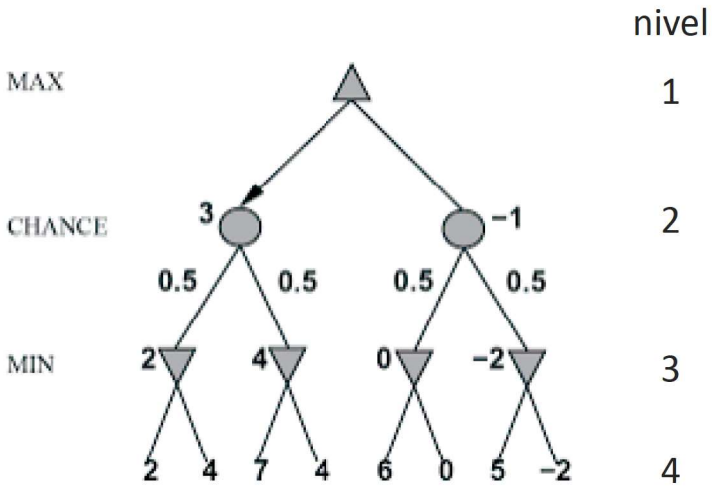


Ilustración 4: Ejemplo del algoritmo Expectiminimax

$$expected\_value = 0.5 * 2 + 0.5 * 4 = 1 + 2 = 3$$

$$expected\_value = 0.5 * 0 + 0.5 * (-2) = -1$$

Búsqueda expectiminimax - Algoritmo

```
función expectiminimax(nodo, profundidad, turno, turnoSiguiente)
1:   si nodo es terminal o profundidad == 1
2:       retornar utilidad(nodo)
3:   si turno es Min entonces
4:      // retornar el valor del nodo hijo con menor valor
5:      α = +∞
6:      para cada hijo del nodo
7:         α = min(α, expectiminimax(hijo, profundidad-1, turnoSiguiente, Max))
8:   sino
9:              si turno es Max entonces
10:                     // retornar el valor del nodo hijo con mayor valor
11:                     α = -∞
12:                     para cada hijo del nodo
13:                             α = max(α, expectiminimax(hijo, profundidad-1, turnoSiguiente, Min))
14:             sino // si turno es CHANGE
15:                     // retornar el valor promedio de todos los nodos hijos
16:                     α = 0
17:                     para cada hijo del nodo
18:                             α = α + (probabilidad(hijo) * expectiminimax(hijo, profundidad-1, turnoSiguiente, change))
19:                     α = α / numero total de hijos del nodo
20:   retornar α
```

Práctica – Poda Alfa-Beta

Fuente: https://www.codingame.com/ide/puzzle/tic-tac-toe

Gracias!