euge4
www.wuolah.com/student/euge4

resumenTema4.pdf

Resumen Tema4+5 Diapos+Libro

2° Inteligencia Artificial

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

INTELIGENCIA ARTIFICIAL

Tema 4

BÚSQUEDA CON ADVERSARIOS: JUEGOS

Eugenia Castilla Fragoso

ÍNDICE

1. Juegos bipersonales con información perfecta

Estas situaciones se estudian y resuelven utilizando la **Teoria de Juegos**.
¿qué es un juego? Es cualquier situación de decisión, caracterizada por poseer una interdependencia estratégica, gobernada por un conjunto de reglas y con un resultado bien definido.

Un juego con **información perfecta** es aquel en el que los jugadores tienen a su disposición toda la información de la situación del juego.

2. Árboles de exploración de juegos

- Un árbol del juego es una representación explícita de todas las formas de jugar a un juego
- Correspondencia entre árboles de juegos y árboles Y/O

2.1. Notación min-max

- MAX: primer jugador
- MIN: segundo jugador
- Nodos MAX y nodos MIN
- Los nodos terminales se etiquetan con V(victoria), D(derrota) o E(empate) desde el punto de vista de MAX

El algoritmo min-max calcula la decisión minimax del estado actual. Usa un cálculo simple recurrente de los valores minimax de cada estado sucesor. La recursión avanza hacia las hojas del árbol, y entonces los valores minimax retroceden por el árbol cuando la recursión se va deshaciendo. El algoritmo minimax realiza una exploración primero en profundidad completa del árbol de juegos. Si la profundidad máxima del árbol es m, y hay b movimientos legales, entonces la complejidad en tiempo del algoritmo minimax es $O(b^m)$.

2.2. Algoritmo STATUS

- si J es un nodo MAX no terminal, entonces STATUS(J)=
  - V si alguno de los sucesores de J tiene STATUS V
  - D si todos los sucesores de J tienen STATUS D
  - E en otro caso
- si J es un nodo MIN no terminal, entonces STATUS(J)=
  - V si todos los sucesores de J tienen STAUTS V
  - D si alguno de los sucesores de J tiene STATUS D
  - E en otro caso

2.3. Nuevo modelo de solución

- los juegos complejos no se pueden resolver ya que es imposible la exploración total hasya la terminación
- Nuevo objetivo: encontrar una buena jugada inmediata
- Importancia de la heurística en el proceso

3. El modelo básico

- Arquitectura percepción/planificación/actuación→ se utilizaría en el concepto en el que un agente deliberativo que tiene que resolver el desplazamiento en un entorno para alcanzar sus objetivos
- Búsqueda con horizonte → delimita el espacio de estados, normalmente se fija una determinada profundidad
- Uso de heurísticas → es una función de evaluación estática que se llama cundo un nodo es final o terminal, permite valorar el estado. Debe valorar tanto lo que es bueno como lo que es malo. Premiar lo bueno y castigar lo que es malo.

3.1. La regla minimax

- El valor de V(J) de un nodo J de la frontera de búsqueda es igual a de su evaluación estática, si no
- Si J(V) es un nodo MAX, entonces su valor V(J) es igual al máximo de los valores de sus nodos sucesores
- Si J es un nodo MIN, entonces su valor V(J) es igual al mínimo de los valores de sus nodos sucesores.
- Algoritmo→
  - si J es un nodo terminal, devolver V(J) = f(J), en otro caso
  - para k = 1,2,...n hacer→
    - generar J_k, el k-ésimo sucesor de J
    - Calcular $V(J_k)$
    - si k = 1, hacer AV(J) ← $V(J_1)$; en otro caso, para k >=2
    - hacer AV(J)← max{AV(J),V(J k)} si J es un nodo MAX o
    - hacer AV(J)← min{AV(J),V(J k)} si J es un nodo MIN
- Devolver V(J) = AV(J)

3.2. Poda alfa-beta

Es una mejora sobre el algoritmo minimax. No se generan todos los nodos hasta el horizonte, por lo que es difícil calcular el número de nodos que se van a generar. En el mejor de los casos se generan: $B^{p/2}$
En el peor de los casos se generan todos los estados que generaría el algoritmo minimax.
Para calcular el valor V(J, alfa, beta), hacer lo siguiente:

1. Si J es un nodo terminal, devolver V(J)= f(J). En otro caso, sean $J_1 ... J_n$ los sucesores de J. Hacer k ← 1 si J es un nodo MAX ir al paso 2; si J es un nodo MIN ir al paso 5
2. Hacer alfa ← max(alfa, $V(J_k$, alfa,beta))
3. Si alfa >= beta devolver beta; si no, continuar
4. si k = b, devolver alfa; si no, hacer k← k+1 y volver al paso 2
5. Hacer beta ← min(beta, $V(J_k$, alfa,beta))
6. Si beta <= alfa devolver alfa; si no, continuar
7. si k = b, devolver beta; si no, hacer k ← k+1 y volver al paso 5

3. Juegos en los que interviene un elemento aleatorio

Es igual que el algoritmo minimax pero se añade una capa de probabilidad