PEC 1 - Representación de problemas

Presentación

Primera PEC del curso de Inteligencia Artificial.

Competencias

En esta PEC se trabajaran les siguientes competencias:

Competencias de grado:

- Capacidad de analizar un problema con el nivel de abstracción adecuado a cada situación y aplicar las habilidades y conocimientos adquiridos para abordarlo y solucionarlo.

Competencias específicas:

- Saber representar las particularidades de un problema según un modelo de representación del conocimiento.

Objetivos

Esta PEC pretende evaluar vuestros conocimientos sobre formalización de problemas y búsqueda básica sobre espacio de estados.

Descripción de la PEC

Mirad el siguiente vídeo, donde un robot juega al *Conecta 4* con un adversario humano:

https://www.youtube.com/watch?v=7uiPhECQtrs

Como muchos de vosotros sabréis, este juego consiste en que 2 adversarios alternativamente van colocando fichas (cada adversario tiene un color) con el objetivo de tener 4 fichas de su color alineadas (en horizontal, vertical o diagonal). Tened en cuenta que, aunque el contexto de esta PEC es un juego, no lo abordamos desde el punto de vista de búsqueda con adversarios; el temario necesario para realizar esta PEC es el indicado en el apartado *Recursos*.

En este contexto, se pide formalizar el problema y responder a las siguientes cuestiones.

1. Explicad vuestra formalización del problema. ¿Qué información habrá en cada estado? ¿Cuántos estados posibles habrá en el grafo de estados? ¿Todos los estados son accesibles desde cualquier estado inicial?

Podemos formalizar el problema con una matriz de elementos del conjunto {A, B, O}. Una lista de siete listas de seis elementos de este conjunto cada una también serviría (cada lista representaría una columna). Una lista de seis listas de siete elementos de este conjunto cada una también serviría (cada lista representaría una fila).

Con esta formalización, inicialmente tenemos que la estructura del juego presenta 42 lugares donde ubicar las fichas. Cada lugar puede presentar 3 valores: {Ficha jugador A (lo representaremos con A), Ficha jugador B (B), No ficha (O)}. Así pues, con esta representación tenemos 3^{42} estados posibles.

Obviamente, no todos los estados son accesibles desde cualquier estado inicial, ya que, en cada paso, no se puede escoger libremente la casilla donde ubicar la ficha, sino que dependemos del estado inicial. Es más, con esta representación hay estados imposibles de acceder, ya que vulnerarían las reglas del juego; hemos de contemplar que cada jugador tira una ficha de manera alternante. Por ejemplo, un estado donde la diferencia de número de fichas de uno y otro jugador es superior a 1 es inaccesible.

No obstante, se podría definir una formalización más compleja donde ya se incluyera la restricción sobre los estados accesibles. En este caso, evidentmente, la explicación anterior sería diferente. Es decir, es importante remarcar que los estados existentes/accesibles dependen de la formalización inicial, y no al revés.

2. ¿Cuántos operadores tendremos? ¿Cuáles serán? ¿Cómo relacionan éstos los estados descritos anteriormente?

Formalmente, podemos ver los operadores como funciones que transforman un estado válido (según vuestra formalización) en otro estado válido, siguiendo las normas impuestas por la formalización del problema. Así, en el caso que nos ocupa, necesitamos pasar de un estado a otro colocando una nueva ficha de un color determinado en alguna de las columnas. Para esto, podemos utilizar tanto operadores parametrizados (por ejemplo, un solo operador que recibe la columna y el color, o un operador por columna que recibe el color) como no parametrizados (en este caso serían 14 operadores, contando las combinaciones columna-color).

Lo importante aquí es que para pasar de un estado a otro necesitamos saber qué columna modificamos, y con qué color lo hacemos.

Incluso el propio operador podría determinar el color de la ficha que toca automáticamente a partir del estado actual donde opera, mirando cuántas fichas de cada color hay, y por tanto necesitando tan solo un índice de columna (un solo parámetro). No obstante, para que esto sea válido, necesitamos establecer, en la formalización del problema, una norma que nos indique que la partida la comienza siempre un determinado color de ficha. De lo contrario, no seríamos capaces ni de avanzar desde el estado inicial vacío.

3. Considerando solo los estados válidos (aquellos donde se puede llegar si respetamos las reglas del juego) y un estado inicial donde la estructura del juego está vacía de fichas, ¿dónde aparecen más nodos: en el grafo de estados del problema, o en el árbol de búsqueda? Considerad también que la partida la inicia siempre el mismo jugador (por ejemplo, el jugador A). Justificad la respuesta.

Teniendo en cuenta que consideramos tan solo los estados válidos, la respuesta depende de si en la creación del árbol se realiza detección de estados repetidos o no. En el primer caso, ambas estructuras tendrán los mismos nodos; en el segundo, el árbol de búsqueda tendrá más nodos, dado que muchos de ellos se repetirán, ya que diferente orden en las acciones tomadas pueden llevar a los mismos estados.

4. Considerando que hacemos una búsqueda no informada, responded a las siguientes cuestiones:

a) ¿Cuál es el factor de ramificación? 7 (una rama por cada columna), excepto en estados avanzados donde hay columnas llenas.

b) Definid el estado del minuto 2:16 (considerad A las fichas del robot y B las del humano) de acuerdo con vuestra formalización del problema.

O	B	O	A	O	B	O
O	A	O	A	A	B	O
O	A	O	B	B	A	O
B	A	O	A	A	B	O
A	B	O	A	B	A	O
B	B	B	A	B	B	O

Considerando una lista de listas que representan las columnas, tendríamos algo como:

((OOOBAB)(BAAABB)(OOOOOB)(AABAAA)(OABABB)(BBABAB)(OOOOOO))

c) Partiendo de este estado, y haciendo una búsqueda en anchura, ¿cuál es la profundidad mínima donde encontramos un nodo donde acaba la partida? ¿Quién gana? Recordad que el primer paso lo da el robot (fichas A).

Profundidad 2. Se da cuando en el primer paso se inserta ficha A en la tercera columna, haciendo posible que a continuación se inserte una ficha B también en la tercera columna, haciendo cuatro B en diagonal. Gana B (el humano).

d) Observad el estado en el minuto 2:22. Tal y como se indica en el vídeo, el robot tiene victoria asegurada, pero decide dar falsas esperanzas al adversario y no ir directamente a por la victoria. Imaginad que vosotros tenéis que programar esta funcionalidad (no ir directamente a por la victoria, pero asegurando que ésta no se escapa). ¿Qué condiciones consideraríais que se han de dar en el árbol de búsqueda para poder hacerlo? ¿Cuántas jugadas se puede estar haciendo esto sin poner en peligro la victoria?
Para que esta funcionalidad se pueda llevar a cabo ha de existir un estado ganador a profundidad 1 aplicando un determinado operador (insertando una ficha en una determinada columna) y, en caso de no hacerlo, si el oponente utilizase el mismo operador (roba la jugada), volviéramos a tener un estado ganador a profundidad 1. Por lo tanto, hemos de explorar el árbol hasta una profundidad 2 siempre. Esto lo podremos estar haciendo todos los turnos que queramos, sempre y cuando no haya ningún estado ganador para el rival a profundidad 2, es decir, que el hecho de escoger nosotros una jugada que no nos da la victoria al momento no implique darle la victoria al oponente en un solo movimiento.