Tema-4.pdf

Blublu__3

Inteligencia Artificial

2º Grado en Ingeniería Informática

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

TEMA 4

4.1 Introducción

Hasta ahora hemos estudiado entornos monoagentes. Ahora vamos a estudiar los entornos multiagentes en los que hay que considerar las acciones de otros agentes y cómo afectan a su propio estado. Hay dos tipos de entornos multiagente:

- Cooperativos: en el que todos los agentes trabajan juntos para alcanzar un objetivo común
- Competitivos: el objetivo de cada agente entra en conflicto con los del resto

En definitiva, vamos a estudiar técnicas de búsqueda con adversario que son problemas de búsqueda en entornos multiagente competitivos que a partir de ahora llamaremos juegos.

Los juegos son laboratorios perfectos para investigar técnicas de resolución de problemas porque el estado de un juego es fácil de representar y el repertorio de acciones que, en general, se puede realizar en un juego es pequeño. Además los resultado de las acciones están definidos y todo esto en contraposición de los juegos físicos. Por otro lado los juegos son duros de resolver.

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. En un juego cada jugador intenta conseguir el mayor beneficio para sus intereses y la solución de un juego es la que nos va a permitir indicar a un jugador qué resultado puede esperar y cómo alcanzarlo.

En este tema nos vamos a centarr en juegos bipersonales y con información perfecta, es decir, estamos en un entorno observable y determinista. Otra de las características de los juegos que vamos a ver en este tema es que son de suma nula juegos en los que la solución final el beneficio de un jugador es total y para el otro jugador es nulo.

4.2 Juegos como problema de búsqueda

Un juego puede definirse como un tipo especial de problema de búsqueda con los siguientes elementos:

- Estado inicial: donde se representa la posición inicial del tablero y se identifica el jugador que mueve.
- Repertorio de acciones: representan movimientos legales asociados a un estado.
- Modelo de transición: representado como una función sucesor que decide por cada estado cuáles son los movimientos legales y el estado resultante.
- Una función de comprobación con función de test terminal que se va a realizar para comprobar que un estado de juego es un estado terminal, es decir, es un estado donde el juego finaliza.
- Una función de valoración o función de utilidad esta función define el valor numérico para cada estado terminal y para cada uno de los jugadores.

El estado inicial, el repertorio de acciones y la función sucesor definen el árbol de juego que es una representación teórica del juego, es decir, no está representado completamente en memoria en el que los nodos son estados del juego y los arcos movimientos. Vamos a llamar al grafo explícito árbol de búsqueda.

4.2.1 Encontrar una solución a un juego

Nuestro objetivo es explorar los suficientes nodos para poder llegar a una decisión aceptable. No buscamos un camino, estamos buscando una estrategia contigente.

Para calcular una estrategia desde el punto de vista de max hay que explorar todos los movimientos posibles que puede hacer min y viceversa. Esta es precisamente la característica de los juegos que nos permiten establecer una correspondencia entre juegos y grafos Y/O.

Mientras que en un problema de búsqueda estándar la solución es una secuencia de movimientos que llevan a un estado objetivo, en los problemas de juegos es una estrategia contingente que tiene en cuenta los movimientos de min y que toma la forma de un árbol Y/O.

Entonces vamos a entender por resolver un juego encontrar una valoración para el nodo inicial,es decir, etiquetar al nodo inicial con un valor que informe de que max gana o pierde, esto se hará a través de sus sucesores y a la vez determinar si hay una estrategia ganadora para min, para max o para ninguno de los dos.

Para encontrar la valoración de sus sucesores descenderemos al siguiente nivel y así sucesivamente hasta que lleguemos a los nodos terminales y en realidad como los nodos terminales tienen la valoración lo que vamos a hacer es propagar la valoración de los nodos iniciales hacia arriba.

4.3 Algoritmo STATUS

Este algoritmo recorre recursivamente todo el árbol de juego y decide en cada nodo si el nodo es de victoria, derrota o empate, desde el punto de vista de max. Lo decide preguntándoselo a cada uno de sus hijos de tal forma que si, un nodo j no terminal, es max STATUS se preguntará si gana o no gana max y lo etiquetará con una V si max gana en alguno de sus sucesores. Los hijos de nodos max eran nodos O. Lo etiquetará con D si todos sus sucesores determinan que max pierde y lo etiquetará con E en otro caso.

Para encontrar la valoración de los sucesores de un nodo max (que son todos nodos min). STATUS bajará a un siguiente nivel y se preguntará recursivamente por todos los sucesores de estos nodos min de tal manera que, si un nodo j es no terminal y min ,se preguntará si gana o no MAX y lo etiquetará como victoria si todos sus sucesores tienen valor de victoria (tener en cuenta que ahora todos los hijos de nodos min son hijos Y) y lo etiquetará con D en el momento en el que encuentre algún hijo a partir del cual pierda max y lo etiquetará con E en otro caso.

Este proceso va a continuar recursivamente hasta que lleguemos a los nodos terminales es el momento en el que se van a propagar hacia arriba los valores de la valoración. Observar que valorar el estado inicial es equivalente a determinar si partiendo del estado inicial hay algun movimiento que me lleve a la victoria.

Reflexión

Ahora que conocemos un procedimiento o un algoritmo para explorar un árbol de un juego y poder determinar si un jugador puede encontrar o no una estrategia contingente para ganar o no un juego. Vamos a parar un momento para hacer una reflexión.

Por un lado hay un resultado teórico a partir de este algoritmo que nos dice que todo juego con información perfecta tiene solución porque como exploramos todo el árbol de juego este algoritmo nos va a garantizar una estrategia en la que todos los nodos terminales son situaciones en la que max ganas entonces el juego se resuelve para max o por el contrario si no encontramos una estrategia ganadora para max el juego se resuelve para min, incluso si hay empate también decimos que el juego se ha resuelto con empate.

Status es un algoritmo que en la práctica no se utiliza porque tiene que recorrer todo el árbol de juego y esto es inviable para árboles grandes. Sin embargo status nos da ideas sobre como proceder para solucionar un juego por un lado STATUS tiene una exploración recursiva de un árbol con backtracking y por otro lleva

QUIERES 15€ ?

TRAE A TU CRUSH DE APUNTES

si juegas con fuego te fuegas

si consigues que suba apuntes, te llevas 15€
+ 5 Wuolah Coins para los próximos sorteos

acabo un proceso de propagación de valores hacia arriba. Estos son elementos importantes pero nos falta uno muy importante que es el de tratar de reducir el espacio de búsqueda mediante heurísitcas y lo vamos a hacer mediante dos heurística una es la valoración de cada estado basándonos en la intuición y la experiencia, es decir, usando valoración heurística numérica y la otra usando una búsqueda con horizonte para reducir el espacio de búsqueda.

4.4 Algoritmo Minimax

A partir de ahora vamos a dejar de tener etiquetas para tener valoraciones numéricas en los nodos y que estos nodos en los que vamos a tnere las valoraciones no son los verdaderos nodos terminales si no que son los nodos en la frontera a una profundidad que hayamos definido y por tanto vamos a tener nodos valorados por una funcion de valoración que decimos que es estática. Cuando hablamos de valoración estática nos referimos a que la hemos calculado directamente de alguna forma. Esta valoraciones no son valoraciones precisas del juego, son una estimación de cómo de prometedoras son essas posiciones que están representadas en los nodos de la frontera.

El algoritmo minimax nos va a decidir es la mejor valoración que podemos obtener ante una situación de juego y por tanto el mejor movimiento que podemos hacer. Es decir minimax nos permite obtener una estrategia óptima para max siempre que asumamos que min sigue una estrategia óptima para derrotar a max.

El algoritmo minimax procede de forma muy similar a STATUS pero en cada nodo aplica una regla de valoración diferente llamada regla minimax así que el valor minimax de cada nodo se define recursivamente y consiste en:

- La valoración estática del nodo si este es terminal.
- El máximo de las valoraciones de sus sucesores si es un nodo máx.
- El mínimo de las valoraciones minimax de sus sucesores si este es un nodo min.

Observar que la principal característica de la regla minimax es que max trata de maximizar su ganancia y min trata de minimizar su pérdida.

Existe también una variante que se llama NEGAMAX que explota la siguiente igualdad:

$$\max(a,b) = -\max(a,b) = -\min(-a, -b)$$

y por tanto, esta variante no calcula el máximo y el mínimo según sea el nodo si no que en su lugar siempre calcula el máximo de sus sucesores pero cambiando el signo en cada llamada recursiva a megamax. Se entiende que esta variación es más sencilla y no modifica el valor devuelto por minimax.

Sobre la complejidad de este algoritmo hay que decir que en el peor de los casos minimax realizara una exploración completa en profundidad del árbol del juego, entonces la complejidad en tiempo es exponencial (Ob^m) siendo b el factor de ramificación y m la profundidad del árbol. La complejidad en espacio es lineal O(m) o O(bm) si guarda los sucesores antes de realizar la llamada recursiva. En cualquier caso, minimax sin limite de profundidad no es practicable para juegos reales.Si llevamos un proceso de búsqueda con profundidad limitada las decisiones serán mejores conforme mayor sea el límite de profundidad.

4.5 Poda alphabeta

Es una técnica que reduce el orden de complejidad del algoritmo minimax manteniendo la correctitud del algoritmo. Que se basa en descartar la exploración de nodos del espacio de búsqueda utilizando dos parámetros uno llamado alpha y otro llamado beta. De manera que si alpha es igual o mayor que beta todos los nodos hermanos se descartarán (se podarán).

Alpha representa el mejor valor encontrado hasta el momento para max, es decir, una variable auxiliar que usan los nodos max para calcular el máximo por tanto su valor va a ser inicialmente -inf y lo vamos a ver como una cota inferior, es decir, una cota que solo puede crecer. Esta variable se actualiza solo con valores resultantes de evaluar los sucesores min de un nodo max.

Beta es el mejor valor encontrado hasta el momento por los nodos min y es una variable auxiliar que vamos a usar en los nodos min para calcular el mínimo por lo tanto el valor inicial será +inf y lo vamos a ver como una cota superior, es decir, una cota que solo puede decrecer. Esta variable se actualiza solo con valores resultantes de evaluar los sucesores max de un nodo min.

Este algoritmo va a ser igual que el algoritmo minimax solo que si se cumple el criterio de poda el valor devuelto para un nodo max es beta y para un nodo min es alpha.

Para poder hacer podas al menos una parte del árbol ha debido de ser explorada usando una búsqueda primero en profundidad. La exploración tiene que ser hasta el límite de profundidad que hayamos decidido porque los valores de alpha y beta que vamos calculando tienen que estar definidos por los valores de los nodos terminales.

Además el número de ramas podadas durante la búsqueda depende del grado en el que alpha y beta se aproximen a los valores finales de cada uno. Es decir si ordenamos los sucesores de un nodo min de menor a mayor y los sucesores de un nodo max de mayor a menor estaríamos ante un caso ideal con O(b^d/2), es decir, que el número de nodos terminales que vamos a explorar con alpha beta en óptima circunstancias a una profundidad es el mismo que exploraríamos ocn un algoritmo minimax con la mitad de profundidad.

En el peor de los casos alpha y beta no produce ningún efecto y en el caso promedio el orden de complejidad es O(b^3/4d) o lo que es lo mismo en el caso de promedio la profundidad de búsqueda puede incrementarse en un 33%.

4.7 El modelo básico para decisiones en tiempo real

Para definir una función de valoración que guíe a un agente hacia posiciones adecuadas hay que tener en cuenta que:

1. la función de evaluación tiene que ordenar bien los estados en los que se gana, después en los que se empata y después en los que se pierde.
2. El tiempo dedicado a calcular ese valor no debe de ser elevado.
3. Para estados no terminales la evaluación debe ser una buena estimación de las posibilidades actuales de ganar o perder.

En la mayoría de las funciones de evaluación se hace primero un análisis de las características a tener en cuenta. Esto nos va a permitir calcular valores numéricos por separado para cada una de las características y luego lo que vamos a hacer es combinarlas para calcular un valor total asignando pesos para cada característica. En todo caso, estos pesos y características dependen mucho de la experiencia humana en el juego y si no la tenemos los pesos de la función de evaluación se pueden estimar con técnicas de machine learning.

Algunos problemas que pueden surgir si decidimos establecer una profundidad constante.

En primer lugar nos podemos encontrar que en la frontera de la profundidad haya posiciones que no sean estables, es decir, estados en los que la evaluación cambie drásticamente si se avanza una sola posición porque estemos a punto de hacer un gran movimiento.Por eso los programas de juegos se aseguran que una posición es estable antes de parar la búsqueda en ese estado Para determinar si una posición es estable se comprueba si uno o dos movimientos más adelante su valoración no varía drásticamente.

Aún usando la técnica anterior, nos vamos a encontrar con situaciones en las que el desastre o el éxito van a estar justo más allá del horizonte de búsqueda por ejemplo hay situaciones en las que inevitablemente max puede ganar pero la búsqueda de posiciones estables hace que min pueda retrasar llegar a esa situación de pérdida. No es fácil definir qué es un buen estado en estas situaciones y de todas formas hay técnicas para evitar este efecto horizonte:

- Búsqueda secundaria que consiste en llevar a cabo una búsqueda sobre el mejor resultado del horizonte para ver si esa posición es estable.
- Descenso iterativo con un límite de tiempo en vez de un límite de profundidad.

En cualquier caso las técnicas de búsqueda que hemos conocido son componentes básicos de cualquier programa de ordenador con éxito en la implementación de juegos bipersonales pero deben complementarse con el uso de otras técnicas.

4.8 Juegos en los que interviene un elemento aleatorio

Son juegos en los que los movimientos que tenemos que hacer dependen de un elemento aleatorio. Para estos casos el modelo de árbol de juego lo vamos a extender con un nuevo tipo de nodo que podemos denominar nodo de tipo dado o azar que nos va a representar el resultado del lanzamiento de un dado. Los nodos de tipo dado representan una acción no determinista y cada arco que sale de un nodo dado se etiqueta con su probabilidad asociada. La valoración de los nodos dado se calcula como el valor esperado o esperanza de la valoración de sus sucesores.

En principio el cambio no es complicado si queremos plantearnos la resolución de un juego estocástico o con factor aleatorio, esto nos va a llevar a generalizar la función del valor minimax determinístico a una función que vamos a llamar minimaxEsperado. Los nodos terminales max y min funcionarán igual que antes pero para los nodos de tipo dado se calcula su valor esperado que va a consistir en la esperanza.

Además en este tipo de juegos la idea de buscar hasta un límite de profundidad y evaluar ahí de forma estática lso nodos sigue siendo válida. Pero donde tenemos que tener cuidado es en la definición de la función de evaluación estática ya que hay que procurar que la valoración que escojamos para nodos intermedios sea coherente con la probabilidad de ganar desde una determinada posición.