Ajuste dinámico de profundidad en el algoritmo $\alpha\beta$ ($DDA\alpha\beta$)

Daniel Micol[1] y Pablo Suau[2]

[1] Escuela Politécnica Superior
Universidad de Alicante
San Vicente del Raspeig
03690 Alicante
dmp18@alu.ua.es

[2] Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
San Vicente del Raspeig
03080 Alicante
pablo@dccia.ua.es

Resumen Desde los inicios de la teoría de computación en juegos, diversas técnicas han sido desarrolladas con el fin de mejorar la calidad de juego de la máquina a la vez que se reduce el tiempo de respuesta. Uno de los algoritmos más extendidos para la búsqueda en juegos es el conocido como $\alpha\beta$, el cuál supone una variante respecto del algoritmo Minimax al introducir podas en el árbol de decisión. Actualmente hay numerosas mejoras aplicables a dicho algoritmo, aunque todas ellas con el mismo inconveniente: poseen un alto grado de incertidumbre y pueden acarrear pérdidas de rendimiento o cálculos erróneos. Proponemos una técnica basada en un ajuste dinámico del nivel de profundidad según el estado de la partida que supone una mejora fiable para el algoritmo $\alpha\beta$. Dicha técnica, a la que hemos dado el nombre de *Dynamic Depth Adjustment on the $\alpha\beta$ algorithm ($DDA\alpha\beta$)*, nunca podrá desembocar en pérdidas de rendimiento debido a que está basada en la suposición del peor caso del algoritmo. La experimentación aportada muestra una mejora de rendimiento y fiabilidad de los resultados en comparación con otros métodos. Además, existe la posibilidad de ser utilizado en combinación con otras técnicas.

Palabras clave: inteligencia artificial, búsqueda en juegos, algoritmos heurísticos.

1. Introducción

El algoritmo $\alpha\beta$ es el método más extendido a la hora de realizar búsquedas en árboles de decisión aplicados a juegos formados a partir de un conjunto de elementos con ordenación total [1]. El objetivo del mismo es obtener la mejor jugada para la máquina, tratando de predecir las respuestas del usuario a sus movimientos para así conseguir la victoria en la partida. Está basado en el algoritmo Minimax, diferenciándose de éste en la inserción de operaciones de poda, lo cuál reduce el número de nodos a evaluar en los árboles de decisión [2]. En el algoritmo $\alpha\beta$ dicho número depende de dos factores: las podas realizadas en el árbol, y la profundidad máxima del mismo.

El primero de estos factores no se puede conocer a priori, sino tan sólo después de haber completado el proceso. Existen técnicas para incrementar el número de podas, como es el caso de la profundización iterativa [3], tablas de refutación [4], búsquedas usando ventanas mínimas [5] y heurísticas [6], pero la mayoría posee cierto grado de incertidumbre que puede producir pérdidas de rendimiento [7]. Todos estos métodos reducen el tiempo de ejecución del algoritmo $\alpha\beta$, descartando, mediante heurísticas adecuadas, ramas que son consideradas poco relevantes para la obtención de la solución. Sin embargo, estas heurísticas pueden ser demasiado restrictivas por lo que la búsqueda podría dejar de ser admisible.

Por otro lado, la profundidad máxima del árbol de decisión también determinará la cantidad de nodos hoja a evaluar. Conforme avanza una partida, el número de posibles movimientos se reducirá, y también lo hará la cantidad de nodos evaluables en el árbol de decisión. Basándonos en este hecho, proponemos aplicar técnicas que aprovechen la mencionada reducción de nodos hoja para incrementar el nivel de profundidad máximo del árbol y así poder obtener un mayor nivel de juego sin aumento del tiempo de respuesta de la máquina.

Este artículo se compone de una primera parte en la que introduciremos la técnica propuesta. En la segunda se explicará el método diseñado, y su aplicación a un juego en concreto será descrita en la tercera sección. Por último mostraremos los resultados en la cuarta parte, así como las conclusiones y trabajos futuros en la quinta y última.

2. Ajuste dinámico de profundidad

Por definición, el número máximo de nodos hoja en un árbol $\alpha\beta$ se corresponderá con la cantidad de posibles combinaciones de jugadas realizadas desde el estado actual de la partida hasta el nivel de profundidad máximo establecido a priori para dicho algoritmo [2]. Por lo general, dicho valor de profundidad se define al principio de la partida y es invariante durante el transcurso de ésta. No obstante, conforme el estado de una partida avanza, el número de posibles combinaciones de jugadas disminuye, y por lo tanto también lo hace el número de nodos hoja del árbol $\alpha\beta$. Puesto que el tiempo de respuesta del algoritmo depende en gran medida de la cantidad de nodos que son evaluados, se deduce que incrementando el nivel de profundidad conforme avanza la partida se podrá mantener constante la cantidad de nodos evaluados en cada turno de la máquina,

siendo dicha cifra similar a la del inicio de la partida. Esto permitirá incrementar el nivel de profundidad sin aumento del tiempo de respuesta respecto al principio de la partida, lo que se traduce en un incremento de la calidad de juego de la máquina.

Tomaremos como base para nuestro método un juego teórico caracterizado por dos jugadores que juegan por turnos y en el que ambos intentan ganar, no existiendo el azar. Para resolver dicho juego se aplica el algoritmo $\alpha\beta$ y suponemos que no hay límites de jugadas de un determinado tipo, es decir, que nunca hay poda de una rama por imposibilidad de jugar en esa posición. Hay que destacar que, a priori, el número de podas del algoritmo $\alpha\beta$ resulta impredecible, y la única forma de conocerlo es recorrer y evaluar los nodos de dicho árbol. Por lo tanto, y puesto que estamos realizando una formalización matemática, partiremos de la suposición de que no se realizará ninguna poda. Por esta razón, en cada nivel del árbol se genera un número exponencial de nodos cuyo factor de exponenciación es constante para todo el árbol y coincide con las posibilidades de movimiento.

Sea ω el nivel de profundidad en el estado inicial de la partida, ψ el nivel de profundidad en el estado actual de la misma, N_ω el número de nodos hoja en el estado inicial de la partida y N_ψ el número de nodos hoja en el estado actual de ésta, se deduce que el número de posibilidades de movimiento en el nivel inicial será $\sqrt[\omega]{N_\psi}$. Por lo tanto, suponemos que $\exists \Delta\omega / \sqrt[\omega]{N_\psi}^{(\omega+\Delta\omega)} = N_\omega \wedge \Delta\omega \in \mathbb{N}$, y podemos deducir lo siguiente:

$$\begin{aligned}
\sqrt[\omega]{N_\psi^{(\omega+\Delta\omega)}} &= N_\omega \\
\Rightarrow N_\psi^{\frac{(\omega+\Delta\omega)}{\omega}} &= N_\omega \\
\Rightarrow \log_{N_\psi} N_\psi^{\frac{(N+\Delta\omega)}{\omega}} &= \log_{N_\psi} N_\omega \\
\Rightarrow \frac{\omega+\Delta\omega}{\omega} \log_{N_\psi} N_\psi &= \log_{N_\psi} N_\omega \qquad (1) \\
\Rightarrow \frac{\omega+\Delta\omega}{\omega} &= \log_{N_\psi} N_\omega \\
\Rightarrow \omega+\Delta\omega &= \omega \log_{N_\psi} N_\omega \\
\Rightarrow \omega+\Delta\omega &= \omega \frac{\ln N_\omega}{\ln N_\psi}
\end{aligned}$$

Finalmente, y puesto que el nivel de profundidad será un número natural, tenemos que:

$$\psi = \omega + \Delta\omega \simeq \left[\omega \frac{\ln N_\omega}{\ln N_\psi}\right] \qquad (2)$$

Como se puede observar, el nivel de profundidad en el estado actual de la partida siempre será mayor o igual que en el estado inicial.

No obstante, el método descrito posee un inconveniente, ya que es incapaz de predecir el número de nodos hoja del árbol generado por el algoritmo $\alpha\beta$, por lo que, aun siendo capaz de proporcionar un nivel de profundidad, éste

siempre será calculado para el peor caso del algoritmo desperdiciándose parte de la capacidad computacional de la máquina. Para solventar este problema se propone a continuación un método alternativo y se muestra cómo podría ser aplicado a un juego concreto.

3. El ajuste dinámico en el juego del *Conecta Cuatro*

En un juego real, las posibilidades de movimiento suelen estar limitadas en algún estado de la partida. Por ejemplo, en el ajedrez un peón no podrá avanzar si existe otra ficha en la siguiente fila y misma columna. Otro ejemplo es el *Conecta Cuatro*[1], donde no se podrá colocar una ficha en una columna que ya esté completa. En estos casos es posible realizar una aproximación más fiable que la descrita anteriormente mediante el cálculo de todas las posibilidades de juego, esto es, del número de nodos hoja que poseerá el algoritmo $\alpha\beta$ sin contar las podas, pero teniendo en cuenta las limitaciones mencionadas.

En algunas ocasiones es posible encontrar una expresión que permita calcular el número de nodos hoja de un árbol $\alpha\beta$ según el estado actual de la partida, pero en la mayor parte de los casos la obtención de dicha expresión no resulta trivial. Para aquellos en los que la expresión para el cálculo del número de nodos hoja sea compleja, es aconsejable realizar una modificación que simule el comportamiento del algoritmo $\alpha\beta$ pero sin evaluaciones. Esto proporcionará una aproximación bastante fiable del número de nodos final, con el acarreo de tiempo de ejecución correspondiente.

A continuación nos centraremos en el caso del juego del *Conecta Cuatro* para mostrar una variante simplificada del método propuesto. Ésta consiste en tener en cuenta tan sólo las columnas no llenas del tablero, sin considerar cuántas fichas se podrán depositar en cada una de ellas. Sea τ el número de columnas del tablero, ξ el número de columnas libres, es decir, el número de posibles columnas en las que podremos poner una ficha, y ω el nivel de profundidad inicial, se deduce que τ^ω es el número máximo de nodos hoja del árbol $\alpha\beta$, esto es, para el peor caso del algoritmo. Conforme se vayan insertando fichas en el tablero, ξ irá decreciendo. Por lo tanto, al avanzar la partida, sería posible aumentar el nivel del árbol evaluando como máximo el mismo número de nodos que en el estado inicial de la partida. Es decir, si conocemos τ^ω, deberíamos encontrar un valor de $\Delta\omega$ tal que $\tau^\omega \geq \xi^{\omega+\Delta\omega}$. El valor de $\Delta\omega$ se puede extraer de forma aproximada tal como se muestra a continuación:

$$\begin{aligned} & \log_\xi(\tau^\omega) \geq \log_\xi(\xi^{\omega+\Delta\omega}) \\ \Rightarrow\ & \log_\xi(\tau^\omega) \geq (\omega+\Delta\omega)\log_\xi \xi \\ \Rightarrow\ & \log_\xi(\tau^\omega) \geq \omega+\Delta\omega \end{aligned} \tag{3}$$

[1] Las características y reglas de este juego pueden consultarse en múltiples páginas en Internet, como por ejemplo http://en.wikipedia.org/wiki/Connect_four.

Debido a que el nivel de profundidad es un número entero, es necesario realizar una aproximación:

$$\begin{cases} \log_{\xi}(\tau^{\omega}) \geq \omega + \Delta\omega \\ \psi = \omega + \Delta\omega \end{cases}$$
$$\Rightarrow \psi \leq \left\lceil \log_{\xi}(\tau^{\omega}) \right\rceil \tag{4}$$
$$\Rightarrow \psi \leq \left\lceil \frac{\ln(\tau^{\omega})}{\ln \xi} \right\rceil$$

A partir del razonamiento anterior, se puede deducir, en cada turno de la máquina, cuánto se podría incrementar la profundidad del árbol de juego sin afectar al rendimiento. Este nuevo nivel se obtendrá a partir del factor de ramificación actual y el inicial del nodo raíz.

Esta técnica propuesta es aplicable a juegos en las que hay pocas restricciones de movimiento al igual que en el *Conecta Cuatro*. No obstante, para aplicarlo a otros juegos en los que las jugadas estén más restringidas (como por ejemplo el ajedrez o el otelo, juegos en los que no se puede colocar una ficha en cualquier posición del tablero), habría que añadir un factor que tuviera en cuenta dichas restricciones.

4. Experimentación y resultados

Todas las pruebas presentadas a continuación fueron realizadas sobre el juego del *Conecta Cuatro* para diferentes dimensiones del tablero. Primero se expondrán los resultados para el método general descrito en el segundo apartado, y posteriormente para el específico de dicho juego desarrollado en el tercer apartado. Las pruebas fueron realizadas mediante una aplicación programada en lenguaje Java, compilada usando JDK 1.4.2 y ejecutada en un Apple iMac G5 a 2 GHz, 512 MB de RAM y Mac OS X Tiger Versión 10.4.4.

En esta sección se muestran dos tipos de resultados, relacionando el nivel de profundidad con el número de fichas en el tablero, y este último valor con el tiempo de respuesta en milisegundos. El número de fichas en el tablero se corresponde con los movimientos ya realizados, y proporciona una idea de cuál es el estado de la partida. Se supone que el juego finaliza cuando se llena el tablero, ya que para obtener un mayor número de medidas hemos forzado un empate entre los dos jugadores del *Conecta Cuatro*, no deteniendo la partida cuando se produjera un cuatro en raya. Además, hay que destacar que el valor del eje de coordenadas de las gráficas satura en 100 para los que muestran el nivel de profundidad y en 50 para los que hacen lo propio con el tiempo de respuesta. Esto es debido a que en el primer caso interesa observar la evolución de la profundidad máxima hasta el final de la partida, pero en cuanto al tiempo de respuesta, llega un momento en el que se estabiliza en un valor muy bajo cercano a cero, por lo que su representación a partir de este punto no aporta datos útiles.

El primer experimento realizado consistió en aplicar la versión general de la técnica propuesta a tableros de diferentes dimensiones para observar su comportamiento en situaciones variadas. Para ello, se optó por emplear tableros de juego cuadrados[2] de dimensiones 7, 8, 9 y 10 respectivamente. Los resultados muestran un notable incremento del nivel de profundidad, tal y como muestra la Figura 1. El incremento mencionado fue de una magnitud similar para todas las dimensiones de tablero, pero se produjeron en estados más avanzados de la partida conforme se aumentó el tamaño del mismo. Esto puede provocar que el aprovechamiento de la mejora producida por el método *DDA*$\alpha\beta$ ocurra cuando la partida ya esté prácticamente decidida y su aplicación sea de poca utilidad. En estos casos habrá que encontrar algún factor que haga incrementar con anterioridad el nivel de profundidad tratando de no comprometer el tiempo de respuesta de la máquina.

A continuación se analizan los tiempos de respuesta para los casos representados en la Figura 1. Tal como se muestra en la Figura 2, dichos tiempos no se vieron afectados significativamente por el incremento de profundidad producido, siendo siempre inferiores a los obtenidos con el nivel inicial de profundidad, por lo que el rendimiento del algoritmo $\alpha\beta$ no se ve perjudicado. Se puede observar un incremento de los tiempos de ejecución de cada jugada hasta el turno número 25 aproximadamente, por lo que este incremento no es producido por el de la profundidad, ya que éste se produce posteriormente. Además, es posible darse cuenta de que, tal y como se afirma en las secciones anteriores, el aumento de la profundidad máximo no repercute negativamente en el tiempo de respuesta de cada movimiento de la máquina.

Posteriormente se repitió la misma experimentación aplicando el método específico descrito para el juego del *Conecta Cuatro*. Su mayor facilidad de diseño e implementación tiene como contrapartida el hecho de que produce los incrementos de profundidad más tardíamente que el método general. Esto queda reflejado en la Figura 3, donde se observa además que los niveles máximos obtenidos tras la aplicación del método específico serán mayores que los producidos por el general. Por último, cabe destacar que el incremento en cuestión sólo se producirá cuando se llene una columna, hecho que es muy variable, por lo que podrá producir mejoras casi imperceptibles según evolucione la partida.

Al igual que en el método general, en el específico el aumento de niveles de profundidad no afecta negativamente al rendimiento del algoritmo $\alpha\beta$. Esto queda reflejado en la Figura 4, donde se observa cómo a partir de los puntos de incremento de niveles representados en la Figura 3 no se produce un aumento del tiempo de ejecución, por lo que la eficiencia del algoritmo no queda comprometida.

Los resultados de la experimentación demuestran que en la mayor parte de los casos el algoritmo *DDA*$\alpha\beta$ específico aplicado al juego del *Conecta Cuatro* utiliza niveles de profundidad mayores que el general, aunque serán muy pocas las jugadas en las que se explote este incremento de nivel ya que se produce muy tardíamente. Además, el método general posee un comportamiento más estable

[2] Un tablero cuadrado es aquél que posee el mismo número de filas que de columnas.

que permite el uso de un nivel de profundidad alto del árbol durante un mayor número de jugadas, aunque no se llegue a una profundidad tan elevada como al aplicar el método específico. En ambos casos el tiempo de ejecución fue muy similar al que se hubiera obtenido si no se hubiera aumentado el nivel del árbol de forma dinámica.

5. Conclusiones y trabajo futuro

Los resultados obtenidos muestran que es posible incrementar de forma dinámica el nivel de profundidad del algoritmo $\alpha\beta$ de una forma segura y sin que conlleve incrementos en el tiempo de respuesta mediante la suposición del peor caso. Este incremento ayudará a la máquina a contemplar un mayor número de futuras jugadas, mejorando su forma de jugar. Además, es posible realizar cálculos estadísticos del nivel de podas medio de un árbol $\alpha\beta$ para reducir el número teórico de nodos a calcular y así realizar incrementos mayores del nivel de profundidad, con el riesgo que ello conlleva si las suposiciones no son ciertas.

El método general propuesto puede ser complejo de calcular e implementar para ciertos juegos con muchas posibilidades de movimiento por ficha, como es el caso del ajedrez. Para ello, se pueden utilizar variantes más sencillas de dicha técnica aplicadas a juegos concretos con la consiguiente pérdida de efectividad. Además, dicho método se basa en el peor caso del algoritmo ya que es incapaz de predecir el número de podas, por lo que en ocasiones no se alcanzará el máximo nivel de profundidad posible para un intervalo de tiempos de respuesta permitido.

Trabajos futuros pueden estar relacionados con cálculos estadísticos para predicciones sobre el número de nodos que evaluará un árbol $\alpha\beta$ para una aproximación más óptima del nuevo nivel de profundidad. Además, se podrán estudiar diversos juegos de mayor complejidad como es el caso del ajedrez para observar el comportamiento del método propuesto.

Referencias

1. M.L. Ginsberg and A. Jaffray. Alpha-beta pruning under partial orders. *MSRI Publications*, 42:37–48, 2002.
2. F. Escolano, R. Rizo, P. Compañ, and M. Á. Cazorla. *Fundamentos de Inteligencia Artificial*. Publicaciones Universidad de Alicante, 1999.
3. J. Gillogly. *Performance Analysis of the Technology Chess Program*. PhD thesis, Department of Computer Science, Carnegie-Mellon University, 1978.
4. S.G. Akl and M.M. Newborn. The principle continuation and the killer heuristic. pages 466–473. ACM Annual Conference, 1977.
5. J. Pearl. Scout: A simple game-searching algorithm with proven optimal properties. First Annual National Conference on Artificial Intelligence, Stanford, 1980.
6. J. Schaeffer. The history heuristic and alpha-beta search enhancements in practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1203–1212, November 1989.
7. R.L. Rivest. Game tree searching by min/max approximation. *Artificial Intelligence*, 34(1):77–96, December 1987.

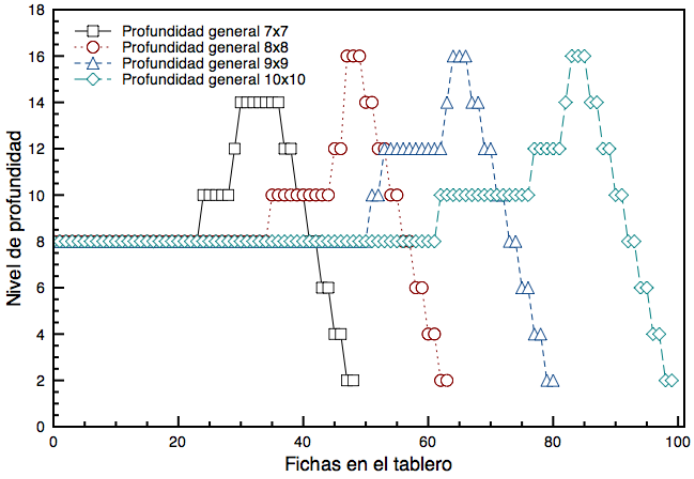

Figura 1. Evolución del nivel de profundidad usando el método general de $DDA\alpha\beta$.

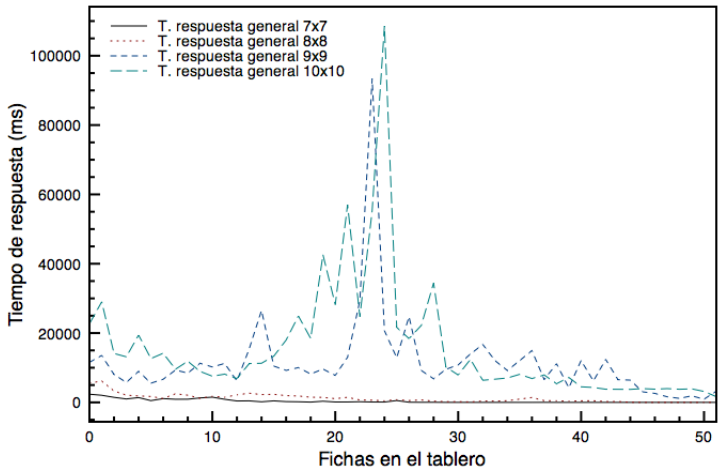

Figura 2. Evolución del tiempo de respuesta usando el método general de $DDA\alpha\beta$.

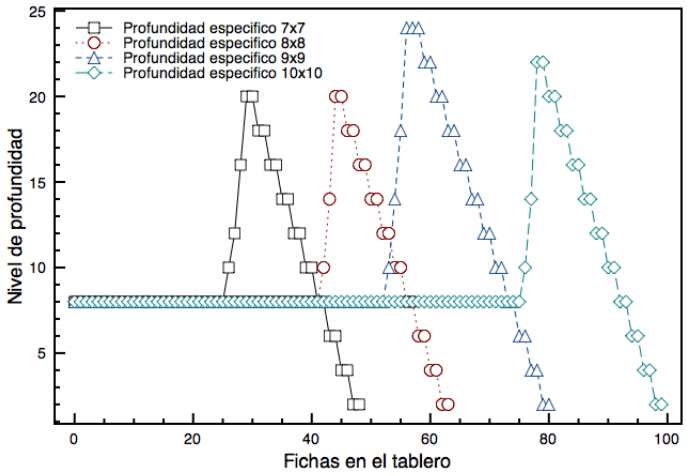

Figura 3. Evolución del nivel de profundidad usando el método específico de $DDA\alpha\beta$.

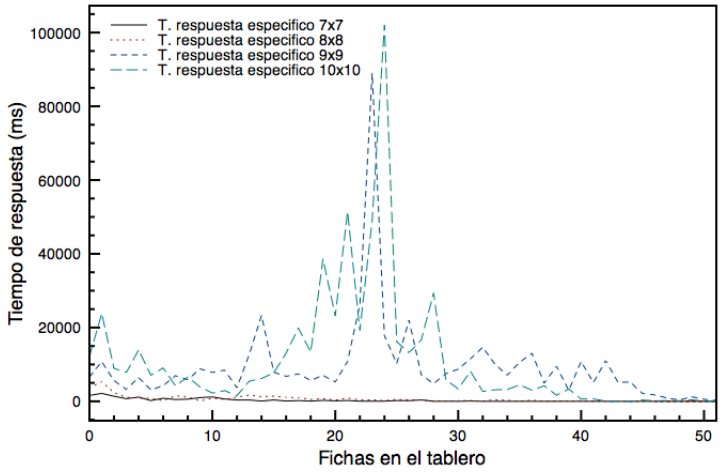

Figura 4. Evolución del tiempo de respuesta usando el método específico de $DDA\alpha\beta$.