|  |
| --- |
| Asignatura: |
| Ingeniería del Conocimiento |

|  |
| --- |
| Título del documento: |
| **Laboratorio 2:**  **Búsqueda con adversarios** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Fecha |
| Grupo: |  | 21 |  | 2/12/18 |
|  |  |  |  |  |
| Miembros: | 1- | Gonzalo de las Heras |  |  |
|  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nombre de fichero: |  | Fecha: |  |  |  |  |
| LAB02-INCO-GRUPO21-MEMORIA.docx **(.pdf)** |  | **2/12/18** |  |  |  |  |

Índice

[1 Resolución de los ejercicios y discusión 4](#_Toc531214620)

[1.1 Primer juego: NIM 4](#_Toc531214621)

[1.1.1 Ejercicio 1. Espacio de Estados 4](#_Toc531214622)

[1.1.1.1 Estado 4](#_Toc531214623)

[1.1.1.2 Operadores 4](#_Toc531214624)

[1.1.1.3 Coste g(n) 4](#_Toc531214625)

[1.1.2 Ejercicio 2. Función de evaluación 4](#_Toc531214626)

[1.1.3 Ejercicio 3. 5](#_Toc531214627)

[2 Segundo juego: Othello/Reversi 8](#_Toc531214628)

[2.1.1 Ejercicio 1. Espacio de Estados 8](#_Toc531214629)

[2.1.1.1 Estado 8](#_Toc531214630)

[2.1.1.2 Operadores 8](#_Toc531214631)

[2.1.1.3 Coste g(n) 9](#_Toc531214632)

[2.1.2 Ejercicio 2. 9](#_Toc531214633)

[2.1.3 Ejercicio 3. 9](#_Toc531214634)

[3 Bibliografía 10](#_Toc531214635)

Índice de ilustraciones

[Ilustración 1. Ejecución con profundidad máxima. Tiempo de cálculo. 6](#_Toc531214604)

[Ilustración 2. Ejecución con profundidad máxima. Tiempo de cálculo. 7](#_Toc531214605)

[Ilustración 3. Tablero inicial Othello. 8](#_Toc531214606)

Índice de tablas

[Tabla 1. Operadores juego NIM. 4](#_Toc531214615)

[Tabla 2. Código Python de la función de evaluación para NIM. 5](#_Toc531214616)

[Tabla 3. Ejecución con profundidad máxima. Jugadas por turno. 6](#_Toc531214617)

[Tabla 4. Ejecución sin profundidad máxima. Jugadas por turno. 7](#_Toc531214618)

[Tabla 5. Operadores juego Othello. 8](#_Toc531214619)

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# Resolución de los ejercicios y discusión

## Primer juego: NIM

### Ejercicio 1. Espacio de Estados

#### Estado

Un estado será representado un número natural, el cual indica el número de fichas restantes.

* El estado inicial será el número de fichas al comienzo del juego (FichasRestantes).
* El estado final es siempre un estado con el número de fichas restantes igual a 0 (FichasRestantes == 0).

#### Operadores

Existen 3 operadores:

|  |  |  |
| --- | --- | --- |
| **Operador** | **Precondición** | **Acción** |
| CogerFicha\_1 | FichasRestantes >= 1 | FichasRestantes -= 1 |
| CogerFicha\_2 | FichasRestantes >= 2 | FichasRestantes -= 1 |
| CogerFicha\_3 | FichasRestantes >= 3 | FichasRestantes -= 1 |

Tabla 1. Operadores juego NIM.

#### Coste g(n)

La función de coste es uniforme (0). Para nuestro juego, este dato no nos afecta.

### Ejercicio 2. Función de evaluación

Para este ejercicio, existe una función de evaluación perfecta y de bajo coste. Dado el número de fichas restantes, podemos saber con seguridad si el jugador al que le toque el turno va a ganar o perder. Con esto asumimos que ambos jugadores van a ejecutar el mejor movimiento posible, es decir, uno no va a perder a propósito.

Código Python:

|  |
| --- |
| def \_\_CalcularGanador(self, Fichas, Tipo):  """  Función para calcular si el jugador actual va a ganar dado un número de fichas.  :param Fichas: Número de fichas.  :param Tipo: Tipo de nodo MIN o MAX.  :return: Devuelve si el jugador va a ganar.  """  if Fichas % 4 == 1:  if Tipo == "MAX":  return -1 # Gana MIN  else:  return 1 # Gana MAX  else:  if Tipo == "MAX":  return 1 # Gana MAX  else:  return -1 # Gana MIN |

Tabla 2. Código Python de la función de evaluación para NIM.

### Ejercicio 3.

**Comparativa de ejecución del juego con/sin restricción de profundidad máxima 2 \* K siendo K = 2.**

Ejecutar el juego sin profundidad máxima lo único que produce, es que los tiempos de cálculo sean muy elevados cuando las fichas restantes son mayores que 15. Esto se debe a que, el programa debe de generar el árbol completamente para evaluar todos las posibles jugadas en cada turno.

Si ejecutáramos el juego con profundidad máxima, es decir, con decisiones imperfectas para evitar el problema del tiempo de cálculo, de la mejor jugada para el jugador máquina, observamos que las jugadas elegidas por el programa son exactamente iguales a la ejecución del juego sin profundidad máxima.

Esto en principio no podría tener sentido, pues se trata de decisiones imperfectas. Sin embargo, al ser la función de evaluación perfecta (sabiendo el número de fichas restantes, sabemos quien va a ganar), el programa no necesita crear el árbol entero para saber con certeza el mejor movimiento en cada turno.

Para demostrar lo anteriormente comentado, ejecutamos el juego con y sin profundidad máxima y número de fichas igual a 20. Comprobamos que los movimientos con decisiones imperfectas sean igual a aquellos tomados cuando se ha generado el árbol completamente.

Con profundidad máxima 2\*K siendo K=2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Turno** | **Jugador** | **Restantes** | **Acción** | **Tiempo de cálculo** |
| 1 | Humano | 20 | 3 | - |
| 2 | Máquina | 17 | 1 | 0.0004260540008544922 s. |
| 3 | Humano | 16 | 3 | - |
| 4 | Máquina | 13 | 1 | 0.0002989768981933594 s. |
| 5 | Humano | 12 | 3 | - |
| 6 | Máquina | 9 | 1 | 0.00031375885009765625 s. |
| 7 | Humano | 8 | 3 | - |
| 8 | Máquina | 5 | 1 | 0.00011992454528808594 s. |
| 9 | Humano | 4 | 3 | - |
| 10 | Máquina | 1 | 1 | 5.698204040527344e-05 s. |
| 11 | Humano | 0 | GANADOR | - |

Tabla 3. Ejecución con profundidad máxima. Jugadas por turno.

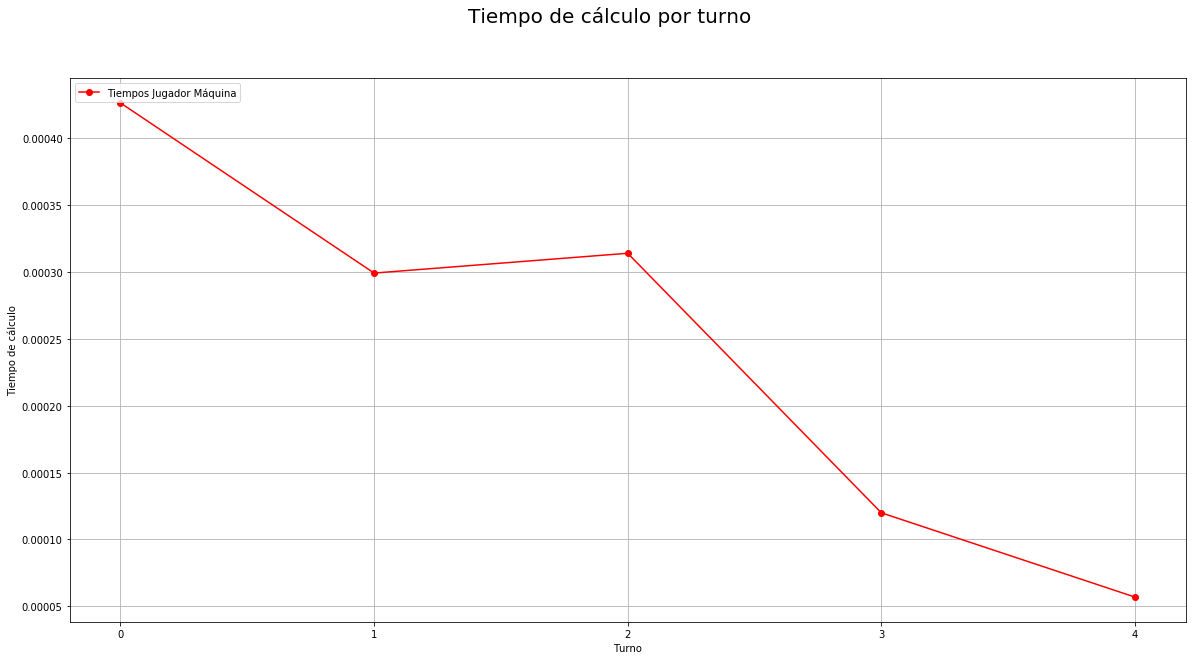


Ilustración 1. Ejecución con profundidad máxima. Tiempo de cálculo.

Observamos que los tiempos de cálculo son extremadamente pequeños, e incluso se van reduciendo más según se encuentran nodos terminales dentro de la profundidad máxima del árbol establecida.

Sin profundidad máxima 2\*K:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Turno** | **Jugador** | **Restantes** | **Acción** | **Tiempo de cálculo** |
| 1 | Humano | 20 | 3 | - |
| 2 | Máquina | 17 | 1 | 56.4747428894043 s. |
| 3 | Humano | 16 | 3 | - |
| 4 | Máquina | 13 | 1 | 0.4095017910003662 s. |
| 5 | Humano | 12 | 3 | - |
| 6 | Máquina | 9 | 1 | 0.0054950714111328125 s. |
| 7 | Humano | 8 | 3 | - |
| 8 | Máquina | 5 | 1 | 0.00015091896057128906 s. |
| 9 | Humano | 4 | 3 | - |
| 10 | Máquina | 1 | 1 | 2.6941299438476562e-05 s. |
| 11 | Humano | 0 | GANADOR | - |

Tabla 4. Ejecución sin profundidad máxima. Jugadas por turno.

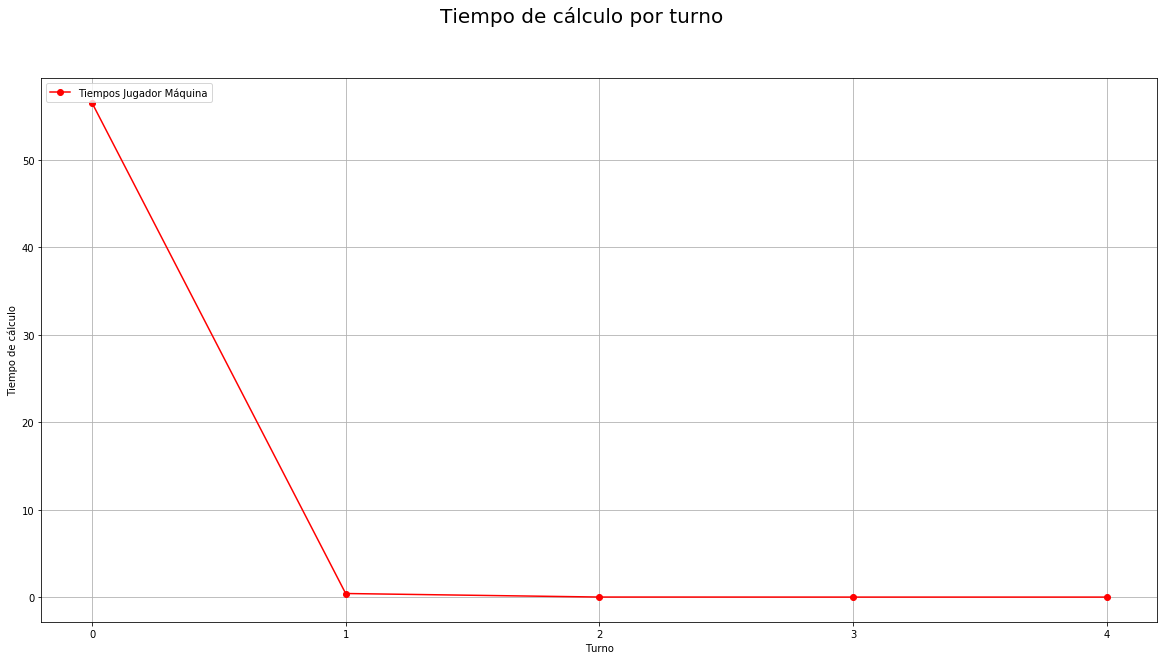


Ilustración 2. Ejecución con profundidad máxima. Tiempo de cálculo.

Observamos que aún mayor número de fichas restantes, los tiempos de cálculo crecen exponencialmente, llegando desde segundos a varios minutos.

**Conclusión**

Minimax con decisiones imperfectas es un buen mecanismo para tomar una decisión aceptable con un coste de tiempo bajo.

Para el caso del juego NIM, minimax con decisiones imperfectas tiene el mismo resultado que minimax generando el árbol completamente.

# Segundo juego: Othello/Reversi

### Ejercicio 1. Espacio de Estados

#### Estado

Un estado será representado una matriz que indica la configuración actual del cuadro del juego (la posición de las fichas en el tablero.

* El estado inicial será el número de fichas al comienzo del juego.

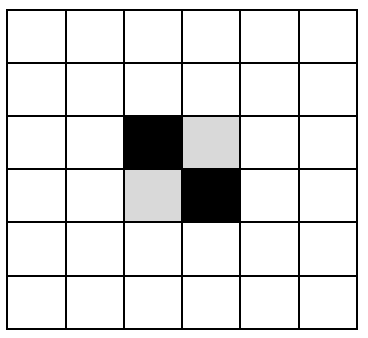


Ilustración 3. Tablero inicial Othello.

* El estado final es aquel en el que ningún jugador puede colocar una nueva ficha, debido a alguno de los siguientes motivos:
  + El tablero está lleno.
  + No existe un movimiento válido para ningún jugador.

#### Operadores

|  |  |  |
| --- | --- | --- |
| **Operador** | **Precondición** | **Acción** |
| ColocarFichaNegra  (x,y) | No hay una ficha en la casilla (x,y). Tablero[x][y] = 0  En esa fila, columna o diagonal hay otra ficha negra que encierra fichas blancas con la que colocamos. | Transforma todas las fichas blancas encerradas en negras. |
| ColocarFichaBlanca  (x,y) | No hay una ficha en la casilla (x,y). Tablero[x][y] = 0  En esa fila, columna o diagonal hay otra ficha blanca que encierra fichas negras con la que colocamos. | Transforma todas las fichas negras encerradas en blancas. |

Tabla 5. Operadores juego Othello.

#### Coste g(n)

La función de coste es uniforme (0). Para nuestro juego, este dato no nos afecta.

### Ejercicio 2.

**¿Influye el orden en el que se analizan los sucesores de un nodo en la eficiencia del algoritmo minimax con poda α-β? Justificar la respuesta con el ejercicio que acabamos de programar.**

Sí que influye. Si se evalúan los hijos de cada nodo en orden creciente o decreciente (dependiendo si es MIN o MAX), se produce la máxima poda. Es decir, se deja de evaluar el mayor número de nodos, ahorrando el mayor tiempo posible.

No obstante, esta situación no se puede forzar. Al necesitar ordenar de manera ascendente o descendente, necesitaríamos saber el valor de minimax. Esto supondría evaluar todos los nodos, justo lo que queremos evitar.

Para nuestro problema, supondría que la máquina podría evaluar muchas más jugadas en menos tiempo. Esto la convertiría en un sistema más competitivo.

### Ejercicio 3.

**Usando la misma profundidad y función de evaluación estática ¿pueden ser distintas las decisiones que tome la máquina en cada jugada usando minimax con poda α-β de las decisiones que tome usando minimax sin poda α-β?**

No, la poda α-β en árboles minimax, produce la misma decisión que el proceso sin usarla, pero más eficientemente. Esto se debe a que la poda α-β solamente deja de evaluar nodos cuando el valor minimax ya está calculado y es invariable.

# Bibliografía

**Australian National University.** Game Trees. [En línea] <https://cs.anu.edu.au/courses/comp1100/lectures/17/17-01-GameTrees.pdf>.

**García-Tejedor, Álvaro.** Moodle UFV. Recursos de la asignatura. [En línea] <http://moodleufv.ufv.es/moodle/>.

**Peters, Jehu.** The Math behind the Magic. [En línea] <https://themathbehindthemagic.wordpress.com/tag/nim/>.

**Ques10.** Apply alpha-Beta pruning on example. [En línea] <http://www.ques10.com/p/13506/apply-alpha-beta-pruning-on-example-given-in-fig-1/>.

**Wikipedia.** Alpha–beta pruning. [En línea] <https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning>.