Inteligencia Artificial: Práctica 4

En esta práctica se pedía implementar heurísticas para el juego "Conecta 4". Para ello, fuimos implementando poco a poco tres heurísticas distintas y probándolas en el torneo propuesto.

Heurística 1

La primera heurística que implementamos corresponde al fichero *jugador1* y básicamente va contando en todas las direcciones posibles viendo qué jugada beneficia más. Se basa en las siguientes funciones:

- *Función Heurística :* es la función principal de la heurística. Si el juego ha acabado devuelve los valores máximos; sino calcula la *puntuacion-actual*, es decir, la puntuación de nuestro jugador, y la puntuación del contrincante *puntuacion-oponente*, con las funciones *puntuacion1* y *puntuacion-contrincante1*. Una vez las calcula, si nuestra puntuación es mayor que el valor absoluto de la puntuación del oponente, devuelve nuestra puntuación, pues quiere decir que nos beneficia más colocar una ficha nuestra que colocar una ficha para molestar a nuestro oponente en una jugada. En otro caso, devuelve la puntuación del contrincante pues quiere decir que es mejor interrumpir una jugada del contrincante para que no gane.

```
(defun heuristica (estado)
 (let* ((tablero (estado-tablero estado))
                (ficha-actual (estado-turno estado))
                (ficha-oponente (siguiente-jugador ficha-actual)))
                      (if (juego-terminado-p estado)
                          (let ((ganador (ganador estado)))               ;; los valores máximo o mínimo
                                    (cond ((not ganador) 0)
                                          ((eql ganador ficha-actual) +val-max+)
                                           (t +val-min+)))
                                   (let ((puntuacion-actual (puntuacion1 tablero ficha-actual))        ;; Calcul
                                         (puntuacion-oponente (puntuacion-contrincante1 tablero ficha-oponente)))
                              (if (> puntuacion-actual (abs puntuacion-oponente))
                                puntuacion-actual                                                    ;; Si favor > abs
                                       puntuacion-oponente)))))     ;; Sino devuelve contra
```

- *Función puntuación1 :* Esta función calcula la heurística a nuestro favor, realiza un bucle para recorrer todas las columnas del tablero contando las fichas consecutivas de cada columna y devolviendo un valor en función del máximo de fichas consecutivas que hay, es decir, calcula las fichas consecutivas con la función *cuenta-fichas-consecutivas1* que devuelve un par formado por el número de ficha máxima y el número de veces que aparece en el tablero. De esta manera, llamando m al número de veces que aparece la ficha, si no hay fichas consecutivas devolvemos 0, si hay una ficha consecutiva, devuelve 20*m, si hay dos fichas consecutivas devolvemos 2000*m, y si hay tres fichas consecutivas devolvemos 200*m. La función devuelve la suma de estos valores en bucle por cada columna.

Lucía Rivas Molina
Daniel Santo-Tomás López

Podemos observar que le damos mayor importancia a cuando hay dos fichas consecutivas, esto se debe a que cuantas más jugadas posibles tengamos, más difícil es para nuestro oponente intentar tirarnos una jugada, de modo que tendremos más posibilidades de juego.

```
'''
(defun puntuacion1 (tablero ficha)
    (let ((punt 0))
      (loop for columna from 0 below (tablero-ancho tablero) do              ;; Bucle pa
            (let*  ((altura (altura-columna tablero columna))
                                (fila (1- altura))                           ;; Ca
                   (maxlist (cuenta-fichas-consecutivas1 tablero ficha columna fila)))
            (setf punt
              (+ punt
                (cond
                    ((= 0 (first maxlist)) 0)
                    ((= 1 (first maxlist)) (* 20 (second maxlist)))
                    ((= 2 (first maxlist)) (* 2000 (second maxlist)))
                    ((= 3 (first maxlist)) (* 200 (second maxlist))))))))  ;; 3 fichas c

                                          punt))                           ;; Suma las
```

- *Función puntuación-contrincante1 :* esta función es similar a la anterior, realiza los mismos movimientos pero para las fichas de nuestro contrincante de modo que lo único que varía son los valores que devolvemos, siendo estos los siguientes:

  - Si tiene cero fichas consecutivas devuelve 0 pues no hay jugada que podamos interrumpir.
  - Si tiene una ficha consecutiva devolvemos -20*m, siendo m el número de veces que aparece una ficha consecutiva.
  - Si tiene dos fichas consecutivas devolvemos -2000*m, dándole mayor importancia a esta opción pues le podemos bloquear las jugadas antes de que las pueda tener.
  - Finalmente si tiene tres fichas consecutivas devolvemos -200*m, para que sea de mayor importancia que cuando tiene una.

```
(defun puntuacion-contrincante1 (tablero ficha)
  (let ((punt 0))
    (loop for columna from 0 below (tablero-ancho tablero) do                ;; Bucle para
          (let*  ((altura (altura-columna tablero columna))
                              (fila (1- altura))                             ;; Maximas t
                 (maxlist (cuenta-fichas-consecutivas1 tablero ficha columna fila)))
          (setf punt
            (+ punt
              (cond
                ((= 0 (first maxlist)) 0)                                    ;; 0 fichas consec
                ((= 1 (first maxlist)) (* -1 (* 20 (second maxlist))))       ;; 1 ficha consecu
                ((= 2 (first maxlist)) (* -1 (* 2000 (second maxlist))))     ;; 2 fichas consec
                ((= 3 (first maxlist)) (* -1 (* 200(second maxlist)))))))))

    punt))
```

- *Función cuenta-fichas-consecutivas1 :* esta función cuenta las fichas consecutivas en todas las direcciones (horizontal, vertical, diagonal ascendente y descendente) y devuelve un par formado por el máximo de ellas y el número de veces que aparece, pues puede ocurrir que aparezca bastantes veces con fichas de por medio.

```
(defun cuenta-fichas-consecutivas1 (tablero ficha columna fila)
    (let* ((horizontal (contar-horizontal tablero ficha columna fila))
           (vertical (contar-vertical tablero ficha columna fila))
           (diag-asc (contar-diagonal-ascendente tablero ficha columna fila))
           (diag-desc (contar-diagonal-descendente tablero ficha columna fila)))
           (maximo (list horizontal vertical diag-asc diag-desc) 0 0)))
```

- *Función máximo :* Esta función recibe una lista, un contador inicializado a 0 y un máximo que inicialmente es 0. Recursivamente va recorriendo la lista y contando el mayor número y las veces que aparece. Esta función nos sirve para saber en qué posición está exactamente la mejor jugada, contando las fichas consecutivas.

```
(defun maximo (lista cont max)
  (cond
    ((null lista) (list max cont))
    (t
      (if (> (first lista) max)
        (maximo (rest lista) 1 (first lista))
        (if (= (first lista) max)
            (maximo (rest lista) (+ 1 cont) max)
            (maximo (rest lista) cont max))))))
```

- Finalmente, hemos utilizado las funciones de contar las fichas en las direcciones horizontal, vertical y diagonal:

```
(defun contar-vertical (tablero ficha columna fila)
  (+ (contar-abajo tablero ficha columna fila)
     (contar-arriba tablero ficha columna (1+ fila))))

(defun contar-horizontal (tablero ficha columna fila)
  (+ (contar-derecha tablero ficha columna fila)
     (contar-izquierda tablero ficha (1- columna) fila)))

(defun contar-diagonal-ascendente (tablero ficha columna fila)
  (+ (contar-abajo-izquierda tablero ficha columna fila)
     (contar-arriba-derecha tablero ficha (1+ columna) (1+ fila))))

(defun contar-diagonal-descendente (tablero ficha columna fila)
  (+ (contar-abajo-derecha tablero ficha columna fila)
     (contar-arriba-izquierda tablero ficha (1- columna) (1+ fila))))
```

Heurística 2

Esta heurística corresponde al fichero *jugador2* y es similar a la heurística anterior, solo que cambiamos los valores devueltos y la forma de calcularlos.

- *Función Heurística :* es la función principal de la heurística y es exactamente igual a la función anterior: si el juego ha acabado devuelve los valores máximos; sino calcula la *puntuacion-actual*, y la puntuación del contrincante *puntuacion-oponente*, con las funciones *puntuacion2* y *puntuacion-contrincante2*. La diferencia más notable entre esta heurística y la del jugador anterior es que esta devuelve la suma de ambas puntuaciones, tanto de la nuestra como la del oponente; mientras que la anterior las devolvía por separado según algunas condiciones. Esto lo hacemos para darle más importancia a la jugada que más heurística tenga, dándole prioridad a la jugada que no nos haga perder.

```
(defun heuristica (estado)
  (let* ((tablero (estado-tablero estado))
              (ficha-actual (estado-turno estado))
              (ficha-oponente (siguiente-jugador ficha-actual)))
                    (if (juego-terminado-p estado)
                  (let ((ganador (ganador estado)))
                           (cond ((not ganador) 0)
                                 ((eql ganador ficha-actual) +val-max+)
                                  (t +val-min+)))
                         (let ((puntuacion-actual (puntuacion2 tablero ficha-actual))
                               (puntuacion-oponente (puntuacion-contrincante2 tablero ficha-oponente)))
              (+ puntuacion-actual puntuacion-oponente)))))
```

- *Función puntuación2 :* Esta función calcula la heurística a nuestro favor, realiza un bucle para recorrer todas las columnas del tablero contando las fichas consecutivas de cada columna y devolviendo un valor en función del máximo de fichas consecutivas que hay, es decir, calcula las fichas consecutivas con la función *cuenta-fichas-consecutivas2* y va sumando ese valor en punt, devolviendo la suma de las heurísticas de todas las columnas. Más adelante veremos que no devuelve un valor aleatorio sino que utilizamos una función dentro de *cuenta-fichas-consecutivas2*..

```
(defun puntuacion2 (tablero ficha)
  (let ((punt 0))
    (loop for columna from 0 below (tablero-ancho tablero) do
      (let*  ((altura (altura-columna tablero columna))
              (fila (1- altura))
              (val (cuenta-fichas-consecutivas2 tablero ficha columna fila)))
                (setf punt
                  (+ punt val))))
    punt))
```

- *Función puntuación-contrincante1 :* esta función es similar a la anterior, realiza los mismos movimientos pero para las fichas de nuestro contrincante de modo que lo

único que varía son los valores que devolvemos, de modo que el valor devuelto es la suma de los valores de todas las columnas multiplicada por -1 de modo que quede negativo.

```
(defun puntuacion-contrincante2 (tablero ficha)
  (let ((punt 0))
    (loop for columna from 0 below (tablero-ancho tablero) do
          (let*  ((altura (altura-columna tablero columna))
                                        (fila (1- altura))
                  (val (cuenta-fichas-consecutivas2 tablero ficha columna fila)))
          (setf punt
            (+ punt (* -1 val)))))
    punt))
```

- *Función cuenta-fichas-consecutivas2 :* esta función cuenta las fichas consecutivas en todas las direcciones (horizontal, vertical, diagonal ascendente y descendente) y devuelve un par formado por la "suma" de ellas, llamando a la función suma. Esta función vuelve a hacer uso de las funciones para contar en las direcciones horizontal, vertical y diagonal.

```
(defun cuenta-fichas-consecutivas2 (tablero ficha columna fila)
  (let* ((horizontal (contar-horizontal tablero ficha columna fila))
         (vertical (contar-vertical tablero ficha columna fila))
         (diag-asc (contar-diagonal-ascendente tablero ficha columna fila))
         (diag-desc (contar-diagonal-descendente tablero ficha columna fila)))
         (suma (list horizontal vertical diag-asc diag-desc))))
```

- *Función suma :* Este es el mayor cambio de esta heurística respecto de la del primer jugador. Antes llamábamos a la función máximo para que calculase el máximo de fichas consecutivas en todas las direcciones, de modo que le dábamos más valor a las direcciones que tenían más fichas consecutivas, ya fueran nuestras o de nuestro contrincante. Esta función recibe la lista formada por las fichas consecutivas en las direcciones horizontal, vertical y diagonal. En función del número de fichas consecutivas que tenga esa dirección, vamos acumulando unos valores u otros en bucle: si tenemos 0 fichas consecutivas sumamos 0, si tenemos 1 ficha consecutiva sumamos 20, si tenemos 2 fichas consecutivas sumamos 2000 dándole mayor importancia para que nos cree más jugadas, y si tenemos 3 fichas consecutivas sumamos 200.

```
(defun suma (lista)
  (let ((elem (first lista)))
    (cond
        ((null lista) 0)
        ((= 0 elem) (+ 0 (suma (rest lista))))
        ((= 1 elem) (+ 20 (suma (rest lista))))
        ((= 2 elem) (+ 2000 (suma (rest lista))))
        ((= 3 elem) (+ 200 (suma (rest lista)))))))
```

En la primera heurística le dábamos mayor importancia a la posición de las fichas, en función de cuántas fichas consecutivas tuviera una dirección, cogíamos el máximo y le dábamos un valor fijo entre 0 y 2000 para que el jugador le diera priorizase las jugadas más beneficiosas para nosotras o las más peligrosas de nuestro contrincante. De modo que siempre intentábamos escoger la mejor jugada.

Esta segunda heurística no se fija en las posiciones sino en las jugadas. A cada jugada le damos un valor suma de todas las fichas consecutivas en todas las direcciones, no solo en la que tenga mayor número de fichas. Esta jugada ayuda a interrumpir las jugadas del contrincante pues ya no sumamos valores fijos, a cada jugada varía pues no son valores estáticos sino dinámicos, de modo que da más juego y más opciones de ganar.

Heurística 3

Esta última heurística está basada en la heurística proporcionada en el enunciado de la práctica. Para mejorarla hemos añadido el cálculo de todas las direcciones posibles: arriba, abajo, izquierda, derecha, arriba-izquierda, arriba-derecha, abajo-derecha y abajo-izquierda. Además, cambiamos los valores de la heurística de modo que le damos más importancia a las jugadas del contrincante pues es mejor interrumpir una jugada que esté a punto de ganar, a mejorar alguna de las nuestras. Probamos con diferentes valores y los mejores eran un término medio. Podemos observar aquí cómo distribuimos los valores. Esta heurística se basa en contar las fichas consecutivas y darle un valor a cada número de fichas consecutivas, según el número de éstas.

```
(setf puntuacion-actual
                        (+ puntuacion-actual
          (cond ((= abajo 0) 0)
                                ((= abajo 1) 10)
                                ((= abajo 2) 100)
                                ((= abajo 3) 1000))
          (cond ((= der 0) 0)
                                ((= der 1) 10)
                                ((= der 2) 100)
                                ((= der 3) 1000))
          (cond ((= izq 0) 0)
                                ((= izq 1) 10)
                                ((= izq 2) 100)
                                ((= izq 3) 1000))
          (cond ((= abajo-der 0) 0)
                                ((= abajo-der 1) 10)
                                ((= abajo-der 2) 100)
                                ((= abajo-der 3) 1000))
          (cond ((= abajo-izq 0) 0)
                                ((= abajo-izq 1) 10)
                                ((= abajo-izq 2) 100)
                                ((= abajo-izq 3) 1000))
          (cond ((= arriba-der 0) 0)
                                ((= arriba-der 1) 10)
                                ((= arriba-der 2) 100)
                                ((= arriba-der 3) 1000))
          (cond ((= arriba-izq 0) 0)
                                ((= arriba-izq 1) 10)
                                ((= arriba-izq 2) 100)
                                ((= arriba-izq 3) 1000)))))
```