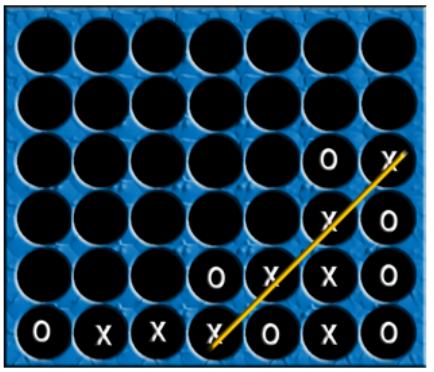
Universidad Complutense de Madrid

Facultad de Informática

CONTACT4



Trabajo realizado por Ana María Martínez Gómez

Métodos algorítmicos en resolución de problemas

2014

Índice

INTRODUCCIÓN

En este trabajo vamos a ver una implementación del contact4 en C++ para poder jugar contra el ordenador o que el ordenador juegue solo. Es decir, veremos una implementación en la que la maquina realiza siempre la mejor jugada. Se utilizará el algoritmo miniMax. También analizaremos el rendimiento añadiendo una función heurística y cortes alpha-beta.

Antes de pasar a la implementación del contact4 vamos a explicar brevemente en qué consiste este juego y a mencionar algunos detalles que serán importantes en la implementación.

Contact4 (conecta4 en español), también conocido como las 4 en línea o las 4 en raya, es un juego de dos jugadores en el que cada jugador pone una ficha de su color por turnos. Gana aquel que primero consiga poner en la misma columna, fila o diagonal 4 fichas de su color seguidas. Típicamente el tablero tiene una dimensión de 6 x 7, aunque en estas dimensiones haciendo la mejor jugada siempre gana el que empieza. Por ejemplo en 7 x 7 el segundo jugador sigue sin poder ganar pero si puede empatar. También consideraremos tamaños más pequeños dado que en algunos casos tarda demasiado en ejecutarse para tableros grandes.

Cabe destacar que en caso de encontrar posiciones igual de buenas, se elegirá cualquiera de las jugadas posibles al azar con igual probabilidad, pues si eligiésemos la primera que encontrásemos estaríamos jugando siempre la misma partida. Para evitar problemas y asegurar que los resultados de las pruebas son siempre los mismos la semilla que se utiliza es siempre la misma. Es decir que si se juegan varias partidas sin cerrar el programa serán distintas, pero si lo cerramos al volverlo a abrir la secuencia será la misma.

Se han implementado varios modos de juego: Humano contra humano, máquina contra humano y máquina contra máquina. En el modo máquina contra humano se podrá elegir si la máquina realiza o no la primera jugada. No se va a comentar nada sobre el modo humano contra humano, dado que no hay que calcular jugadas. El código utilizado puede verse en contact4.cpp, aunque en la siguiente sección se comentan los detalles más importantes.

También se proporciona una versión final del juego pensada para jugar con ella de la que se explican más detalles en el comentario final.

IMPLEMENTACIÓN Y COSTE

LLEGANDO A LOS ESTADOS TERMINALES

En la primera versión no se implementó función heurística ni cortes alpha-beta. Al llegar a una posición ganadora (4 fichas iguales en la misma fila, columna o diagonal) se devuelve 100, al llegar a un empate (nadie ha ganado y no quedan más sitios donde poner) se devuelve 0 y en caso de posición perdedora -100.

Se muestra a continuación el tiempo empleado para calcular cada jugada en una ejecución en un tablero de 5 x 4. No se muestran ejecuciones con tableros más grandes porque tarda demasiado en ejecutarse. El tiempo está expresado en segundos y para cada jugada se eligió entre 0 y 4 jugadas posibles.

Jugada	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Jugador	1	2	1	2	1	2	1	2	1	2
Tiempo	481.675	128,072	22,762	8,599	2,659	0,764	0,151	0,075	0,002	0,003

Como se ha mencionado en el párrafo anterior no es viable jugar con tableros más grandes e incluso en este tamaño los tiempos obtenidos para las primeras jugadas son muy elevados. Veremos en la siguientes secciones como la heurística y los cortes alpha-beta reducen estos tiempos significativamente, aunque sin poder asegurar que se ganará siempre. Observamos que los tiempos se reducen drásticamente en cada jugada, cosa que no va a ocurrir cuando fijemos una profundidad máxima.

CON FUNCIÓN HEURÍSTICA

Vamos ahora a evaluar la posición mediante una función heurística antes de llegar a los estados terminales. Es decir que para buscar cada jugada vamos a permitir una profundidad fija máxima (normalmente par para que hayan jugado el mismo número de veces los dos jugadores) y evaluar la posición en función de las piezas del tablero sin llegar a un empate o a una posición ganadora o perdedora.

Usar funciones heurísticas no nos asegura la victoria, pues para toda heurística podemos encontrar alguna situación para la que no funcione. Aunque no podamos asegurar estar haciendo la mejor jugada, estableciendo una profundidad considerable será mucho mejor que cualquier humano que no puede explorar tantas opciones y además para tableros grandes podremos terminar la partida sin esperar horas a que termine.

Se han implementado tres funciones heurísticas. En todas ellas se ha tratado de fomentar un número razonable de empates en los valores devueltos por las heurísticas, pues como ya hemos comentado en la introducción sino hiciésemos esto estaríamos jugando siempre la misma partida. Esto no solo puede resultar aburrido, sino que además el contrario podría obtener ventaja, dado que tras varias partidas podría saber qué vamos a jugar. Vamos a continuación a explicar en qué consiste cada una de las heurísticas y luego a compararlas para elegir la que consideremos mejor.

heurística 1

La primera heurística hemos tratado que sea lo más sencilla posible y como consecuencia la posición se evaluará muy rápido. Para ello nos vamos a centrar en la importancia de poner las fichas lo más centradas posible, pues de esta forma habrá más formas de conseguir las 4 en raya que si ponemos en las esquinas, donde las filas y las diagonales están más limitadas.

La forma de saber exactamente el número de fichas de cada color que hay en el centro es contarlas, pero explorar el tablero para hacerlo nos llevaría demasiado tiempo. Como queremos que sea sencilla vamos a utilizar solo la altura de las columnas, obteniendo un número entre -99 y 99, donde el valor absoluto de las posiciones más altas sea para aquellas posiciones con las columnas centrales más llenas. Se devolverá un valor positivo en el caso de que la última jugada para llegar a esa posición fuese del jugador 1 y uno negativo si fuese del jugador 2.

El valor que se devuelve depende de M. Por ejemplo para M=7 el valor es

2*cima[M/2]+(cima[M/2-1]+ cima[M/2+1])+(cima[M/2-2]+cima[M/2+2])/2

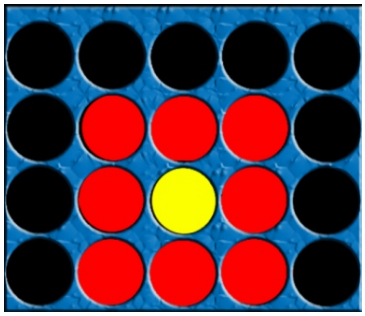
Donde cima[i] es la altura de la columna i. Es evidente que se producirán bastantes empates, algo que ya hemos comentado que nos interesa. Hay que tener en cuenta que este valor está acotado por (2+2+1)N = 5N. Por tanto si 5*N≥100, es decir N≥20 Este valor podría ser mayor que 99 y en este caso dividimos entre N/10 quedando el valor acotado siempre por 50.

Para otros valores de M es similar. El código puede consultarse en contact4.cpp

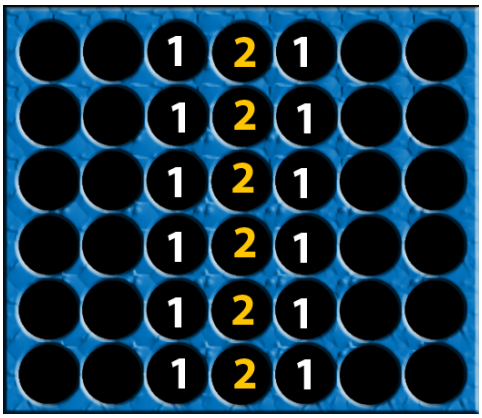
Al ser una heurística tan sencilla cabe esperar que se equivocará muchas veces, pero veremos más adelante al compararla con otras heurísticas más complicadas que su utilización puede ser discutible y sus resultados son bastante buenos.

heurística 2

Esta heurística es la más compleja de las tres. Como el objetivo del juego es juntar fichas del mismo color se darán puntos por tener fichas iguales contiguas, lo que implica que exploraremos todo el tablero. Para cada ficha del jugador 1, se sumarán dos puntos por cada ficha contigua que sea de su mismo color y 1 punto por cada hueco contiguo. Análogo para el jugador 2 pero restando. Se entenderán como fichas contiguas también las que estén en la diagonal, por ejemplo en la siguiente imagen las fichas rojas son todas contiguas a la amarilla:



También se mantendrá la idea de que es mejor poner en el centro que usábamos en la heurística 1. Para ello se sumarán puntos por cada ficha del jugador 1 en el centro y se restarán por cada ficha del jugador 2. Por las fichas en la columna o columnas (si M es par cogemos las dos centrales) central se darán 2 puntos. Si M > 6 se dará además otro punto adicional por cada ficha en cada columna contigua a la/s columnas centrales. La situación se ilustra a continuación:



Para ahorrar tiempo al explorar el tablero, lo haremos por columnas desde la cima de cada una de ellas, dado que encima son todas 0. Además los extremos del tablero (la primera y la última fila y la primera y la última columna) no son muy importantes, pues si queremos potenciar que se ponga en el centro no dar puntos por estas puede hacer la diferencia. Así que no haremos las comprobaciones descritas antes en los extremos porque eso nos ahorra muchas comprobaciones que se traducen en un ahorro de tiempo, además hemos comprobado experimentalmente que esto no afecta a los resultados obtenidos de forma significativa.

Como a la jugada ganadora le hemos dado un valor de 100, debemos asegurarnos de que este valor esté acotado por 99. Por tanto como para cada ficha el valor a sumar es como mucho 2*8, como mucho la mitad de la fichas son del mismo color y se dan como mucho 4*N puntos en las columnas centrales, la puntación está acotada por

(2*8*fichasPuestas)/2 + 4*N = 8*fichasPuestas + 4*N

Nótese que como la profundidad máxima es fija el número de fichas puestas al evaluar la posición será el mismo. Dividiremos la puntuación entre este valor y multiplicaremos por 99. Si el valor no es entero, tomamos solo su parte entera. Esta cota es una cota demasiado alta, por lo que en la mayoría de casos las puntuaciones obtenidas serán mucho menores a 99, lo que fomenta los empates.

heurística 3

En la heurística anterior nos centrábamos en el número de fichas seguidas que tenía cada jugador, lo que no es acertado en muchos casos. Por ejemplo un jugador puede tener tres fichas seguidas, pero que no le sirven para poder hacer 4 en línea y estaríamos dándole muchos puntos sin que sea una buena posición.

Nos centramos ahora en las fichas que tenemos alrededor de cada casilla donde se puede poner. Daremos 2 puntos por cada ficha del jugador 1, y -2 por cada ficha del jugador 1. De esta forma solo tenemos que explorar como mucho M casillas y no exploramos las columnas en las que no se puede poner. Si la casilla está en la primera o la última columna como mucho podemos sumar 2*4 puntos y si está en otra como mucho 2*7 puntos.

También vamos a dar puntos por las fichas en el centro. Se dará 1 punto o -1 punto por cada ficha en la banda central. La banda central tendrá 2 o 3 columnas en función de si M es o no par. Por tanto la suma estará acotado por:

$$16+14*(M-2) +3N$$

Dividiremos la puntación obtenida entre este valor y multiplicaremos por 99, con el fin de no obtener un valor superior a 99 y de que haya empates en las evaluaciones de las posiciones.

Al igual que la heurística anterior también hay posiciones en las que va a fallar. Por ejemplo en las posiciones en las que se tengan muchas fichas seguidas con las que se pueda conseguir 4 en raya pero encima de ellas haya otras fichas.

La ventaja de esta heurística con respecto a la segunda es que tarda menos en calcularse, pero para las tres heurísticas podemos encontrar situaciones en las que la valoración obtenida no sea muy acertada. Pasamos a compararlas.

Comparación

Para todas las pruebas vamos a activar los cortes alpha-beta, pues entendemos que dado que queremos que el juego vaya lo más rápido posible, en una implementación final siempre estarán activados. En la siguiente sección se mostrará el tiempo de la mejor de las heurísticas sin cortes.

A continuación se muestran los tiempos obtenidos en una ejecución para calcular las primeras jugadas con distintas profundidades máximas (con 10 y 12, con más de 12 no es viable jugar con ninguna de las heurísticas porque va demasiado lento) en un tablero 7 x 7 jugando la máquina contra sí misma para cada una de las heurísticas (ambos jugadores con la misma heurística en cada ejecución), también se muestra el número de jugadas entre las que se eligió.

Prof Max = 10

HEURISTICA 1

```
Jugada 1 (jugador 1): 0.402s
Jugada 2 (jugador 2): 0.46s
Jugada 3 (jugador 1): 0.397s
Entre 2 jugadas en la jugada 4
Jugada 4 (jugador 2): 0.489s
Entre 2 jugadas en la jugada 5
Jugada 5 (jugador 1): 0.424s
Entre 4 jugadas en la jugada 6
Jugada 6 (jugador 2): 0.676s
Entre 3 jugadas en la jugada 7
Jugada 7 (jugador 1): 0.894s
Entre 4 jugadas en la jugada 8
Jugada 8 (jugador 2): 0.695s
Entre 2 jugadas en la jugada 9
Jugada 9 (jugador 1): 0.644s
Entre 3 jugadas en la jugada 10
Jugada 10 (jugador 2): 0.321s
Entre 2 jugadas en la jugada 11
Jugada 11 (jugador 1): 0.282s
Entre 2 jugadas en la jugada 12
Jugada 12 (jugador 2): 0.518s
Entre 4 jugadas en la jugada 13
Jugada 13 (jugador 1): 0.513s
Jugada 14 (jugador 2): 0.312s
```

HEURISTICA 2

```
Entre 5 jugadas en la jugada 1
Jugada 1 (jugador 1): 1.2s
Jugada 2 (jugador 2): 1s
Entre 2 jugadas en la jugada 3
Jugada 3 (jugador 1): 2.239s
Jugada 4 (jugador 2): 3.11s
Entre 2 jugadas en la jugada 5
Jugada 5 (jugador 1): 1.039s
Entre 3 jugadas en la jugada 6
Jugada 6 (jugador 2): 0.789s
Jugada 7 (jugador 1): 0.931s
Jugada 8 (jugador 2): 1.291s
Entre 3 jugadas en la jugada 9
Jugada 9 (jugador 1): 0.972s
Jugada 10 (jugador 2): 0.658s
Jugada 11 (jugador 1): 0.494s
Jugada 12 (jugador 2): 0.886s
Entre 2 jugadas en la jugada 13
Jugada 13 (jugador 1): 0.537s
Jugada 14 (jugador 2): 0.404s
Jugada 15 (jugador 1): 0.202s
Jugada 16 (jugador 2): 0.179s
Jugada 17 (jugador 1): 0.089s
Jugada 18 (jugador 2): 0.245s
```

HEURISTICA 3

```
Entre 2 jugadas en la jugada 1
Jugada 1 (jugador 1): 1.473s
Entre 2 jugadas en la jugada 2
Jugada 2 (jugador 2): 1.818s
Jugada 3 (jugador 1): 1.778s
Entre 2 jugadas en la jugada 4
Jugada 4 (jugador 2): 0.925s
Jugada 5 (jugador 1): 0.697s
Jugada 6 (jugador 2): 0.681s
Jugada 7 (jugador 1): 0.686s
Entre 3 jugadas en la jugada 8
Jugada 8 (jugador 2): 0.555s
Entre 2 jugadas en la jugada 9
Jugada 9 (jugador 1): 0.87s
Jugada 10 (jugador 2): 0.78s
Jugada 11 (jugador 1): 0.257s
Jugada 12 (jugador 2): 0.419s
Jugada 13 (jugador 1): 0.645s
Entre 4 jugadas en la jugada 14
Jugada 14 (jugador 2): 1.684s
Entre 4 jugadas en la jugada 15
Jugada 15 (jugador 1): 1.186s
Entre 2 jugadas en la jugada 16
Jugada 16 (jugador 2): 0.905s
```

Prof Max = 12

```
Entre 3 jugadas en la jugada 1
Jugada 1 (jugador 1): 5.678s
Jugada 2 (jugador 2): 6.368s
Entre 2 jugadas en la jugada 3
Jugada 3 (jugador 1): 3.798s
Entre 3 jugadas en la jugada 4
Jugada 4 (jugador 2): 5.324s
Entre 3 jugadas en la jugada 5
Jugada 5 (jugador 1): 4.012s
Entre 2 jugadas en la jugada 6
Jugada 6 (jugador 2): 2.085s
Entre 2 jugadas en la jugada 7
Jugada 7 (jugador 1): 3.375s
Jugada 8 (jugador 2): 0.445s
Entre 2 jugadas en la jugada 9
Jugada 9 (jugador 1): 3.96s
Entre 3 jugadas en la jugada 10
Jugada 10 (jugador 2): 4.018s
Jugada 11 (jugador 1): 11.7s
Jugada 12 (jugador 2): 0.481s
Jugada 13 (jugador 1): 2.139s
Entre 4 jugadas en la jugada 14
Jugada 14 (jugador 2): 1.068s
Entre 7 jugadas en la jugada 15
Jugada 15 (jugador 1): 0.295s
```

```
Entre 2 jugadas en la jugada 1
Jugada 1 (jugador 1): 11.676s
Jugada 2 (jugador 2): 11.683s
Entre 2 jugadas en la jugada 3
Jugada 3 (jugador 1): 9.948s
Entre 2 jugadas en la jugada 4
Jugada 4 (jugador 2): 21.913s
Entre 2 jugadas en la jugada 5
Jugada 5 (jugador 1): 87.045s
Entre 2 jugadas en la jugada 6
Jugada 6 (jugador 2): 14.211s
Jugada 7 (jugador 1): 30.005s
Jugada 8 (jugador 2): 10.391s
Jugada 9 (jugador 1): 34.353s
Jugada 10 (jugador 2): 8.719s
Jugada 11 (jugador 1): 18.618s
Entre 3 jugadas en la jugada 12
Jugada 12 (jugador 2): 6.615s
Jugada 13 (jugador 1): 7.935s
Jugada 14 (jugador 2): 8.492s
Jugada 15 (jugador 1): 15.108s
Jugada 16 (jugador 2): 12.778s
Jugada 17 (jugador 1): 32.173s
Entre 2 jugadas en la jugada 18
Jugada 18 (jugador 2): 15.615s
```

```
Entre 3 jugadas en la jugada 1
Jugada 1 (jugador 1): 16.087s
Jugada 2 (jugador 2): 16.285s
Jugada 3 (jugador 1): 14.434s
Entre 6 jugadas en la jugada 4
Jugada 4 (jugador 2): 21.461s
Jugada 5 (jugador 1): 18.562s
Entre 6 jugadas en la jugada 6
Jugada 6 (jugador 2): 10.649s
Jugada 7 (jugador 1): 9.805s
Entre 5 jugadas en la jugada 8
Jugada 8 (jugador 2): 10.109s
Entre 5 jugadas en la jugada 9
Jugada 9 (jugador 1): 17.11s
Entre 5 jugadas en la jugada 10
Jugada 10 (jugador 2): 16.751s
Entre 3 jugadas en la jugada 11
Jugada 11 (jugador 1): 15.994s
Entre 4 jugadas en la jugada 12
Jugada 12 (jugador 2): 18.599s
Entre 3 jugadas en la jugada 13
Jugada 13 (jugador 1): 10.337s
Jugada 14 (jugador 2): 14.224s
Jugada 15 (jugador 1): 3.853s
Jugada 16 (jugador 2): 14.429s
```

Observamos que los tiempos obtenidos son similares en la heurística 2 y 3, a pesar de que es más costoso calcular la heurística 2, esto puede deberse a la cantidad de nodos que se podan con una u otra heurística. Además los tiempos en estas dos heurísticas

son demasiado elevados, lo que hace que no sea viable jugar con ellas con una profundidad mayor que 10. En cambio, aunque con la heurística 1 también aumenta el tiempo al aumentar la profundidad máxima, aun sigue siendo posible mantener una partida con una profundidad de 12. Es además esta profundidad la más alta que lo hace viable pues con profundidad máxima de 14 los tiempos obtenidos son demasiado altos para la heurística 1 también.

Para saber qué heurística es mejor hemos puesto a jugar a un jugador con cada heurística y ver el porcentaje de veces que gana cada jugador o se empata. Se realizaron 10 ejecuciones. Se ha realizado el experimento con un tablero de 6x7, donde realizando la mejor jugada siempre debería ganar el jugador 1, y con un tablero 7 x 7, donde se debería empatar. La profundidad máxima se ha fijado en 10, porque es una profundidad razonable (para valores mayores las heurísticas 2 y 3 no son viables) y en la que se usa la heurística bastantes veces.

6x7 profMax = 10

Jugador1	**H1**	**H1**	**H2**	**H2**	**H3**	**H3**
Jugador2	**H2**	**H3**	**H1**	**H3**	**H1**	**H2**
Gana jugador 1	0%	60%	80%	90%	20%	10%
Empate	10%	10%	0%	10%	0%	10%
Gana jugador 2	90%	30%	20%	0%	70%	80%

Con los datos obtenidos en un tablero 6x7 podemos concluir que la heurística 2 es la mejor y la 3 es la peor. Veamos si la situación es la misma es un tablero 7x7.

7x7 profMax = 10

Jugador1	**H1**	**H1**	**H2**	**H2**	**H3**	**H3**
Jugador2	**H2**	**H3**	**H1**	**H3**	**H1**	**H2**
Gana jugador 1	20%	80%	100%	70%	30%	70%
Empate	10%	0%	0%	0%	10%	0%
Gana jugador 2	70%	20%	0%	30%	60%	30%

En el tablero 7X7 la 2 es mejor que la 1, que es mejor que la 3. Pero al comparar la 2 con la 3 no podemos concluir nada, pues están muy igualadas.

Teniendo en cuenta los resultados de estas dos últimas tablas y los tiempos, vamos a desechar la heurística 3. Esta heurística es similar a la 2 en 7x7 y peor en 6x7 y en tiempo son similares y es peor a la 1 en ambos casos con un tiempo mucho peor.

En cuanto a las heurísticas 1 y 2, la 2 es mejor en esta profundidad, pero al ver los tiempos vimos que era posible aumentar un poco más la profundidad con la heurística 1, cosa que no es posible con la 2. Por tanto vamos a hacer una última prueba con profundidad máxima 10 para el jugador que utilice la heurística 2 y profundidad máxima 12 para el que utilice la 1. Se hará en 7x7 por ser más grande que 6x7. Las partidas ganadas por cada realizando 10 ejecuciones son las siguientes:

Tamaño tablero	**7x7**	**7x7**
Jugador1	**H1 (profundidad máx 12)**	**H2 (profundidad máx 10)**
Jugador2	**H2 (profundidad máx 10)**	**H1 (profundidad máx 12)**
Gana jugador 1	40%	90%
Empate	20%	10%
Gana jugador 2	40%	0%

Observamos que cuando el jugador 1 juega con la heurística 1 hay un empate entre ambos (se ha mejorado el resultado del jugador 1 al aumentar la profundidad), pero al empezar el jugador de la heurística 2 este obtiene una gran ventaja. Los tiempos de la primera ejecución en cada caso son:

Tablero 7x7 empezado H1

```
Entre 3 jugadas en la jugada 1
Jugada 1 (jugador 1): 5.577s
Jugada 2 (jugador 2): 0.984s
Entre 2 jugadas en la jugada 3
Jugada 3 (jugador 1): 11.238s
Jugada 4 (jugador 2): 3.04s
Jugada 5 (jugador 1): 3.334s
Entre 3 jugadas en la jugada 6
Jugada 6 (jugador 2): 0.773s
Entre 2 jugadas en la jugada 7
Jugada 7 (jugador 1): 2.989s
Jugada 8 (jugador 2): 1.294s
Entre 4 jugadas en la jugada 9
Jugada 9 (jugador 1): 17.376s
Jugada 10 (jugador 2): 5.269s
```

Tablero 7x7 empezado H2

```
Entre 5 jugadas en la jugada 1
Jugada 1 (jugador 1): 1.225s
Jugada 2 (jugador 2): 6.321s
Entre 4 jugadas en la jugada 3
Jugada 3 (jugador 1): 0.371s
Entre 3 jugadas en la jugada 4
Jugada 4 (jugador 2): 3.997s
Entre 2 jugadas en la jugada 5
Jugada 5 (jugador 1): 1.766s
Jugada 6 (jugador 2): 3.738s
Jugada 7 (jugador 1): 1.53s
Entre 2 jugadas en la jugada 8
Jugada 8 (jugador 2): 9.506s
Jugada 9 (jugador 1): 1.732s
Entre 3 jugadas en la jugada 10
Jugada 10 (jugador 2): 9.383s
```

A pesar de que en la mayoría de las ejecuciones los tiempos fueron similares, en algunas los tiempos para el jugador de la heurística 1 fueron excesivamente grandes. Esto se debe a los cortes, que para algunas jugadas no son igual de buenos. Esto sumado a que si empieza el jugador de la heurística 2 este gana con un porcentaje muy alto hace que terminemos rechazando la heurística 1 y nos quedemos con la heurística 2 con una profundidad máxima de 10.

CON CORTES ALPHA-BETA

Hemos concluido que usando cortes alpha-beta la mejor heurística es la 2, por tanto vamos a usar esta heurística para comparar los tiempos al ejecutar el programa con y sin cortes. Veamos el resultado de varias ejecuciones en un tablero 6 x 7. Vimos en la sección anterior que la profundidad máxima con cortes era 10, pero sin cortes tarda más de 10 minutos en calcular la primera jugada a esa profundidad, lo que es inviable. Vamos a considerar por tanto profundidad máxima 6 y 8.

Profundidad máxima = 6

Sin cortes

```
Entre 5 jugadas en la jugada 1
Jugada 1 (jugador 1): 0.421s
Jugada 2 (jugador 2): 0.37s
Entre 2 jugadas en la jugada 3
Jugada 3 (jugador 1): 0.296s
Jugada 4 (jugador 2): 0.173s
Entre 2 jugadas en la jugada 5
Jugada 5 (jugador 1): 0.343s
Jugada 6 (jugador 2): 0.127s
Jugada 7 (jugador 1): 0.526s
Entre 2 jugadas en la jugada 8
Jugada 8 (jugador 2): 0.504s
Entre 2 jugadas en la jugada 9
Jugada 9 (jugador 1): 0.413s
Entre 2 jugadas en la jugada 10
Jugada 10 (jugador 2): 0.319s
Jugada 11 (jugador 1): 0.381s
Entre 2 jugadas en la jugada 12
Jugada 12 (jugador 2): 0.304s
Jugada 13 (jugador 1): 0.436s
Jugada 14 (jugador 2): 0.43s
Jugada 15 (jugador 1): 0.119s
Jugada 16 (jugador 2): 0.415s
Entre 2 jugadas en la jugada 17
Jugada 17 (jugador 1): 0.247s
```

Con cortes

```
Entre 5 jugadas en la jugada 1
Jugada 1 (jugador 1): 0.024s
Jugada 2 (jugador 2): 0.014s
Entre 2 jugadas en la jugada 3
Jugada 3 (jugador 1): 0.02s
Jugada 4 (jugador 2): 0.009s
Entre 2 jugadas en la jugada 5
Jugada 5 (jugador 1): 0.012s
Jugada 6 (jugador 2): 0.007s
Jugada 7 (jugador 1): 0.01s
Entre 2 jugadas en la jugada 8
Jugada 8 (jugador 2): 0.013s
Entre 2 jugadas en la jugada 9
Jugada 9 (jugador 1): 0.013s
Entre 2 jugadas en la jugada 10
Jugada 10 (jugador 2): 0.013s
Jugada 11 (jugador 1): 0.011s
Entre 2 jugadas en la jugada 12
Jugada 12 (jugador 2): 0.017s
Jugada 13 (jugador 1): 0.01s
Jugada 14 (jugador 2): 0.014s
Jugada 15 (jugador 1): 0.007s
Jugada 16 (jugador 2): 0.01s
Entre 2 jugadas en la jugada 17
Jugada 17 (jugador 1): 0.012s
```

Profundidad máxima = 8

Sin cortes

```
Entre 3 jugadas en la jugada 1
Jugada 1 (jugador 1): 18.015s
Jugada 2 (jugador 2): 18.106s
Entre 2 jugadas en la jugada 3
Jugada 3 (jugador 1): 13.984s
Entre 2 jugadas en la jugada 4
Jugada 4 (jugador 2): 6.913s
Entre 2 jugadas en la jugada 5
Jugada 5 (jugador 1): 14.859s
Jugada 6 (jugador 2): 4.71s
Entre 3 jugadas en la jugada 7
Jugada 7 (jugador 1): 20.541s
Jugada 8 (jugador 2): 19.68s
Entre 2 jugadas en la jugada 9
Jugada 9 (jugador 1): 15.113s
Jugada 10 (jugador 2): 7.157s
Jugada 11 (jugador 1): 2.4s
Entre 2 jugadas en la jugada 12
Jugada 12 (jugador 2): 2.185s
Jugada 13 (jugador 1): 0.844s
Jugada 14 (jugador 2): 0.251s
Jugada 15 (jugador 1): 0.447s
Jugada 16 (jugador 2): 0.981s
Jugada 17 (jugador 1): 0.367s
Jugada 18 (jugador 2): 0.648s
```

Con cortes

```
Entre 3 jugadas en la jugada 1
Jugada 1 (jugador 1): 0.11s
Jugada 2 (jugador 2): 0.086s
Entre 2 jugadas en la jugada 3
Jugada 3 (jugador 1): 0.13s
Entre 2 jugadas en la jugada 4
Jugada 4 (jugador 2): 0.156s
Entre 2 jugadas en la jugada 5
Jugada 5 (jugador 1): 0.117s
Jugada 6 (jugador 2): 0.07s
Entre 3 jugadas en la jugada 7
Jugada 7 (jugador 1): 0.089s
Jugada 8 (jugador 2): 0.135s
Entre 2 jugadas en la jugada 9
Jugada 9 (jugador 1): 0.109s
Jugada 10 (jugador 2): 0.092s
Jugada 11 (jugador 1): 0.071s
Entre 2 jugadas en la jugada 12
Jugada 12 (jugador 2): 0.081s
Jugada 13 (jugador 1): 0.054s
Jugada 14 (jugador 2): 0.064s
Jugada 15 (jugador 1): 0.024s
Jugada 16 (jugador 2): 0.084s
Jugada 17 (jugador 1): 0.018s
Jugada 18 (jugador 2): 0.065s
```

Estos resultados demuestran la importancia de los cortes, pues sin ellos nos veríamos obligados a reducir la profundidad máxima, lo que provocaría que la máquina jugase peor.

Veamos también el resultado en una ejecución con tablero de 5 x 4, que fue el tamaño que usamos en la primera ejecución (sin heurística). Utilizaremos una profundidad máxima de 12 al ser el tablero más pequeño.

Sin cortes

```
Entre 2 jugadas en la jugada 1
Jugada 1 (jugador 1): 18.258s
Jugada 2 (jugador 2): 13.616s
Entre 4 jugadas en la jugada 3
Jugada 3 (jugador 1): 9.208s
Jugada 4 (jugador 2): 5.528s
Entre 2 jugadas en la jugada 5
Jugada 5 (jugador 1): 1.876s
Jugada 6 (jugador 2): 0.269s
Jugada 7 (jugador 1): 0.181s
Entre 2 jugadas en la jugada 8
Jugada 8 (jugador 2): 0.037s
Entre 4 jugadas en la jugada 9
Jugada 9 (jugador 1): 0.008s
Entre 3 jugadas en la jugada 10
Jugada 10 (jugador 2): 0.001s
Entre 3 jugadas en la jugada 11
Jugada 11 (jugador 1): 0.002s
Jugada 12 (jugador 2): 0s
Entre 3 jugadas en la jugada 13
Jugada 13 (jugador 1): 0.001s
Entre 3 jugadas en la jugada 14
Jugada 14 (jugador 2): 0.001s
Entre 3 jugadas en la jugada 15
Jugada 15 (jugador 1): 0.001s
```

Con cortes

```
Entre 2 jugadas en la jugada 1
Jugada 1 (jugador 1): 0.126s
Jugada 2 (jugador 2): 0.087s
Entre 4 jugadas en la jugada 3
Jugada 3 (jugador 1): 0.13s
Jugada 4 (jugador 2): 0.093s
Entre 2 jugadas en la jugada 5
Jugada 5 (jugador 1): 0.062s
Jugada 6 (jugador 2): 0.027s
Jugada 7 (jugador 1): 0.019s
Entre 2 jugadas en la jugada 8
Jugada 8 (jugador 2): 0.005s
Entre 4 jugadas en la jugada 9
Jugada 9 (jugador 1): 0.002s
Entre 3 jugadas en la jugada 10
Jugada 10 (jugador 2): 0.004s
Entre 3 jugadas en la jugada 11
Jugada 11 (jugador 1): 0.003s
Jugada 12 (jugador 2): 0s
Entre 3 jugadas en la jugada 13
Jugada 13 (jugador 1): 0.002s
Entre 3 jugadas en la jugada 14
Jugada 14 (jugador 2): 0.003s
Entre 3 jugadas en la jugada 15
Jugada 15 (jugador 1): 0.002s
```

Vemos que el tiempo se ha reducido drásticamente, aunque como ya comentamos se ha hecho a costa de no poder asegurar estar haciendo la mejor jugada.

COMENTARIO FINAL

A lo largo del trabajo comprobamos la importancia de la heurística, que hace posible jugar en tableros grandes en un tiempo razonable. También comparamos distintas heurísticas e hicimos diversas pruebas para decidir cuál era la mejor.

También se comprobó lo necesarios que son los cortes alpha-beta, que hacen que podamos analizar las mismas jugadas en un tiempo mucho menor.

Teniendo en cuenta todo lo mencionado en el trabajo se realizó una implementación del contact4 con la heurística 2 (elegida como la mejor) y habilitando los cortes alpha-beta con el fin de que funcione lo más rápido posible y el usuario no tenga que estar esperando. Esta versión está pensada para jugar con ella y tiene varios niveles de juego. La diferencia entre los distintos niveles es la profundidad máxima, en el más fácil es 2 y en el más difícil es 10 (hemos comprobado que no puede ser mayor que 10). El código de esta versión se encuentra en el archivo contact4Juego.cpp