ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

ESTUDIO DE TÉCNICAS HEURISTICAS PARA JUGAR AL 'GO'

TÍTULO DEL PROYECTO FIN DE CARRERA

Alumno: Fernando Noáin Fernández

Tutor: Miguel Pagola Barrio

Pamplona, Fecha de defensa. 15-7-2010

Índice

Prólogo

Con este proyecto se pretende desarrollar el algoritmo “mini-max” y el algoritmo “mini-max con poda alfa-beta" para el juego “Go”, el cual es un juego de mesa estratégico para dos personas. Una vez hecho esto, se estudiará la diferencia de nodos evaluados entre ambos algoritmos para diferentes profundidades, con el propósito de mostrar la mejoría que se obtiene incorporando la técnica de la poda alfa-beta al algoritmo mini-max propiamente dicho. También se creará y probará una función de evaluación basada en técnicas heurísticas para el mismo juego, con el objetivo final de maximizar la eficiencia del algoritmo diseñado, es decir, que la máquina sea lo mejor posible jugando a dicho juego.

Fernando Noáin Fernández

Capítulo 1. Introducción

1.1.- Concepto “GO”

1.1.1. ¿Qué es?

El “GO” es un juego de estrategia para dos personas creado en china hace más de 2500 años. El objetivo del mismo es intentar cercar territorios dentro del tablero del juego. El go es rico en complejas estrategias a pesar de sus simples reglas.

El juego se realiza por dos jugadores que alternativamente colocan piedras sobre las intersecciones libres del tablero, este tablero es una cuadrícula de 19 líneas verticales por 19 líneas horizontales, formado 361 intersecciones. Los novatos suelen jugar en tableros más pequeños de 13*13 o de 9*9, sin otros cambios en las reglas del juego. Sobre las intersecciones se colocan alternativamente las piedras que son negras o blancas.

El objetivo del juego es controlar una porción más grande del tablero que el oponente. Una piedra o grupo de estas, se captura y retira del juego si no tiene intersecciones vacías adyacentes, es decir, si se encuentra completamente rodeada por piezas del color contrario. Ubicar piedras juntas ayuda a protegerlas entre sí y evitar ser capturadas. Pero ponerlas separadas haces que tengas mayor influencia sobre mas porción del tablero, entonces, hay que saber jugar con estas dos variables, haciendo que nuestro control sobre el tablero sea el mayor posible, pero que nuestras piezas están lo mayor de protegidas posibles.

En Japón existen enormes escuelas que son como universidades para enseñar Go, de hecho es una forma de vida, los que valen para el GO se dedican a jugar campeonatos (que tienen premios millonarios), dar conferencias, asistir a eventos o enseñar. Estas personas son muy respetadas y veneradas. Los japoneses aplican el Go a todo, desde la propia vida a los negocios, y de hecho un buen nivel de habilidad en el juego es muy apreciado a la hora de conseguir trabajo o plaza en una buena universidad (el nivel de go es un dato fundamental en el currículum). Como curiosidad, se dice que los altos mandos americanos en la segunda guerra mundial, aprendieron a jugar al GO para tratar de entender los movimientos de las tropas japonesas (el Go es una disciplina militar obligatoria en Japón).

1.1.2. Equipo para empezar a jugar

- **Tablero:** El tablero, comúnmente llamado goban, por su nombre en japonés. Mide entre 45 y 48 centímetros de largo y entre 42 y 44 de ancho. Los tableros chinos son ligeramente más largos, debido a que sus piedras también son ligeramente más largas. El tablero japonés tradicional tiene entre 10 y 18 centímetros y es de suelo.

- **Piedras**: Un equipo de piedras de Go, usualmente está formada por 181 piedras negras y 180 blancas, dado que una cuadrícula de 19*19 tiene 361 puntos. Hay una piedra negra de más, porque el jugador negras es el que empieza la partida. Hay dos tipos principales de piedras, las piedras convexas, en el que uno de los lados es plano, y biconvexas, en el que ambos lados tiene una curvatura similar. Las piedras convexas puestas por el lado plano, tienden menos a moverse, mientras que las biconvexas, son más fáciles de retirar del tablero a la hora de comer o de terminar la partida. En China el juego normalmente se lleva a cabo con piedras convexas, mientras que en Japón con las biconvexas.

- **Tazones**: Se podría jugar sin ellas, simplemente tienen la utilidad de guardar las fichas que aún no se han puesto en juego. Hay un tazón para cada jugador.

1.1.3. Historia

Según algunas leyendas, el Emperador Yao (2337-2258 a. C.) solicitó a su consejero Shun que diseñara un juego que enseñara disciplina, concentración y equilibrio a su hijo Dazhu, quien se supone era desjuiciado. Otras teorías sugieren que el juego fue inventado por generales y jefes del ejército chino, quienes usaban piedras para señalar posiciones de ataque en mapas, o que el equipo usado actualmente para el juego fue alguna vez usado para realizar lecturas de la suerte.

El Go originalmente se jugaba sobre una cuadrícula de 17x17, pero ésta se modificó a un tamaño de 19x19 durante la Dinastía Tang (618–907).

En China, el juego fue considerado el juego popular de la aristocracia. El Go era considerado una de las Cuatro Artes Tradicionales de los eruditos chinos, junto con la caligrafía, la pintura y la interpretación del instrumento musical guqin.

El Go fue llevado a Japón y Corea en algún momento entre los siglos V y VII d. C., donde se volvió popular en las clases altas. En Corea, el juego evolucionó en una variante llamada Sunjang baduk en el siglo XVI. El Sunjang baduk fue la versión que más se jugó en el país hasta finales del siglo XIX.

1.2. Juegos en I.A

1.2.1. Introducción a juegos en I.A

El juego competitivo ha estado presente desde los inicios del hombre. Se han inventado juegos de todo tipo, ya sean físico o mentales, como el fútbol, dardos, cartas etc.

Los juegos de estrategia se encuentran entre los que más apasionan al ser humano. Precisamente uno de los juegos que más apasionan en gran parte en Europa es el ajedrez, o podemos hablar del Go si nos dirigimos hacia el continente Asiático.

Con la aparición de los computadores modernos, y su rápida evolución en potencia de cálculo, se han creado nuevos juegos de estrategia, así como también se han simulado estos juegos en el computador, de tal forma que este tenga cierta inteligencia, y puedas competir contra él. En gran parte es lo que vamos a hacer en el proyecto, simular el juego Go en el ordenador, y que este sea suficientemente inteligente para poder competir contra él.

La inteligencia Artificial es una rama de la informática que trata de enfocar el concepto de inteligencia en las máquinas.

La I.A es bastante utilizada en la teoría de juegos, sobre todo porque nos da la posibilidad a un conjunto infinito de evaluaciones, cosa que no la hacia la Investigación Operativa que era la encargada anteriormente.

En el campo de los Juegos, nos podemos encontrar con diferentes categorías de problemas, por ejemplo una de las propiedades a tener en cuenta para poder resolver el problema:

- **El número de jugadores necesarios para jugar**, información con gran importancia para poder elegir el algoritmo a diseñar.

- Otra de las prioridades a tener en cuenta en estos juegos es saber el **orden de los movimientos**, si los jugadores mueven alternativamente o por azar.

- Una vez tengamos claro el número de jugadores y el orden de los movimientos, hay que tener en cuenta el **conocimiento de los jugadores**, es decir los jugadores pueden tener conocimiento perfecto, donde no afecta el azar y todos saben en todo momento lo mismo, o bien conocimiento imperfecto, donde el azar sí puede participar y además hay ocultación de información entre jugadores.

1.2.2. Métodos utilizados en I.A para juegos

Para los juegos unipersonales, como el solitario o el 8-puzzle suelen utilizarse los algoritmos más comunes de búsqueda heurística, aunque realmente las soluciones a un juego unipersonal pueden encontrarse con algoritmos de fuerza bruta sin heurística alguna. De entre los algoritmos de búsqueda heurística podemos utilizar el A*, que implementa una búsqueda de el primero el mejor (best first). Este algoritmo utiliza una función de evaluación que es la siguiente: f = g + h donde f será el valor estimado del coste total del camino, h(heurístico) es un valor estimado del coste para llegar del estado actual al estado final u objetivo, g es un coste real del llegar del estado inicial al estado actual.

Para juegos con adversario como el Ajedrez, 3 En Raya o el Go, podemos destacar el algoritmo mini-max, al cual le podemos incorporar mejoras como la poda alfa-beta. Este algoritmo es el utilizado en este proyecto para poder resolver el problema en Go, y será explicado más adelante.

Capítulo 2. Reglas para jugar al Go

2.1.- Reglas básicas para jugar al go

2.1.1. Algunas Terminologías

Libertad: es una intersección vacía adyacente a una piedra del tablero. El número de libertades de una piedra varía dependiendo de su posición en el tablero y del hecho de estar o no en contacto con la con otras piedras.

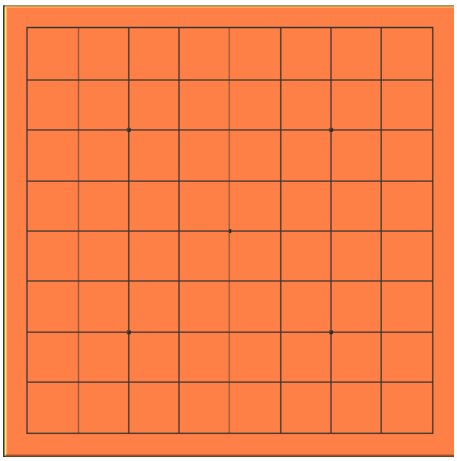
Ojo: El ojo es la fortaleza más impenetrable que existe en el Go. Cuando un jugador posee una formación de fichas que le permite tener un espacio vacío en el interior, se formará un ojo, el ojo es el punto clave que no se puede atacar al rival, y que permite a la fortaleza mantenerse, ya que colocar una ficha en este sitio sería suicidio.

Grupo: Es un conjunto de fichas del mismo color conectadas entre sí.

Atari: Cuando a una piedra le que una única libertad, decimos que está en atari, es decir, en peligro de ser capturada.

2.1.2. Inicio de la partida

Al inicio de la partida el tablero (en nuestro caso de 9x9) el tablero se encontrará vacío, después ambos jugadores juegan alternadamente.



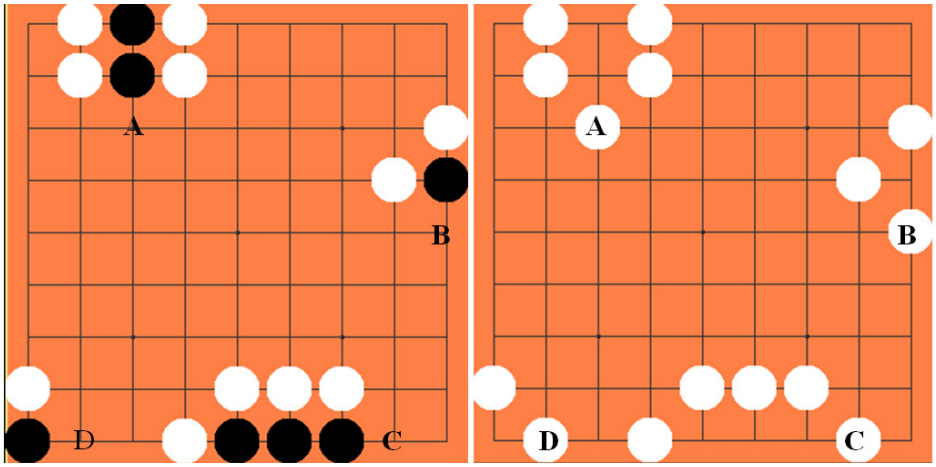
La primera ficha será puesta por el jugador que tiene las fichas negras, y la colocara en cualquiera de las intersecciones del tablero, luego le tocara mover al jugador con fichas blancas.

En cada jugada el jugador que tiene el turno, solo podrá poner una única piedra, que ya no se moverá, a no ser que sea capturada.

En caso de empate ganan las blancas, porque han empezado negras.

2.1.3. Captura

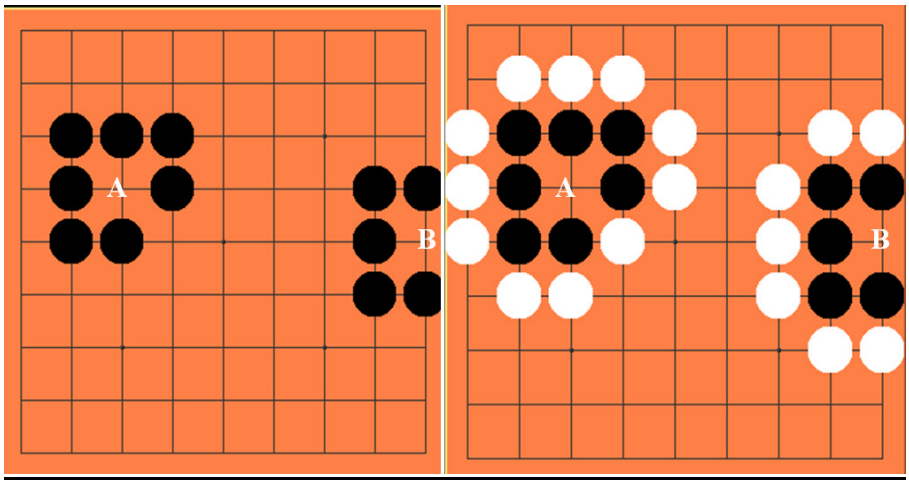
Cuando un jugador hace una jugada que priva de su última libertad a una piedra o formación del oponente debe sacar las piedras rodeadas del tablero y guardarlas separadas a las demás hasta el final, ya que serán contadas como capturadas.

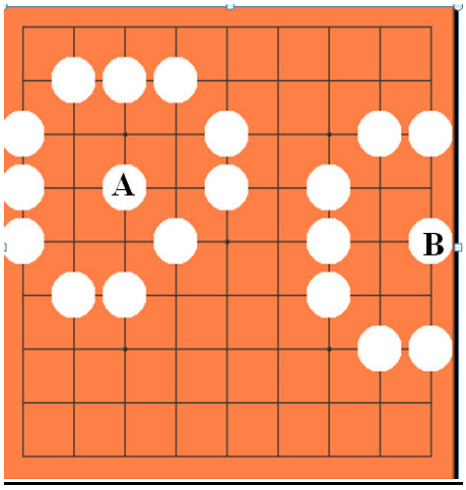


2.1.4. Suicidio

No está permitido hacer una jugada ocupando la última libertad en el interior de una formación enemiga (suicidio) a no ser que esta jugada capture una o más piedras enemigas.

Ejemplo: En la primera gráfica Blancas no puede mover ni en A ni en B porque sería suicidio, pero en la segunda gráfica si podría mover en A y B porque comerían todo el grupo de negras que tienen encerradas, quedando el tablero como en la tercera imagen.

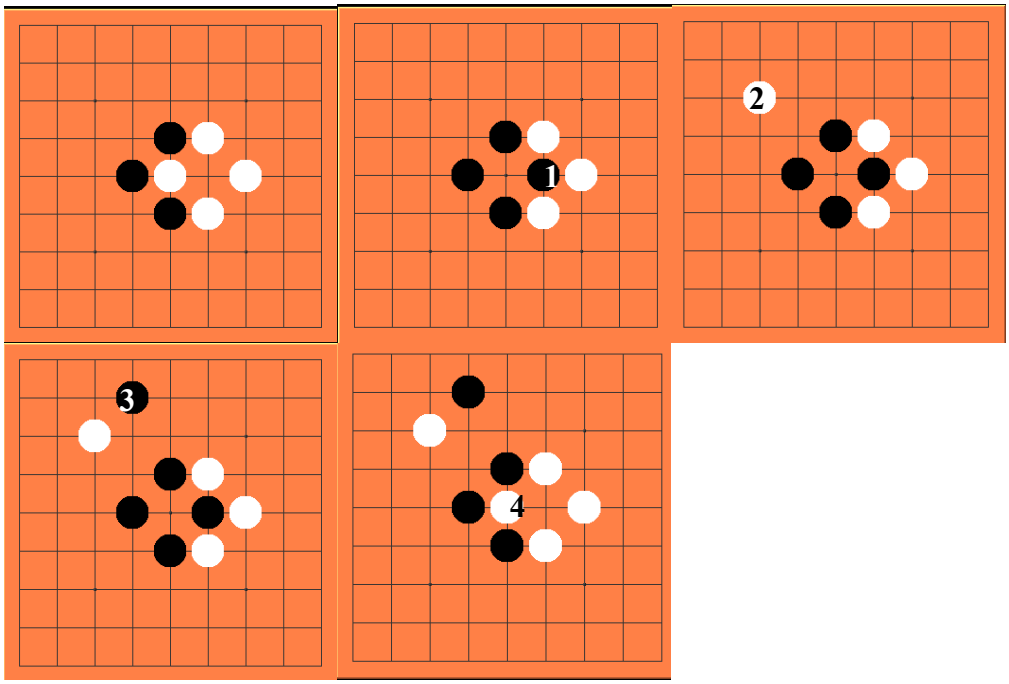




2.1.5. Regla del ko

Cuando una posición del tablero queda consecutivamente en situación de atare por ambos jugadores, el jugador que ha perdido el control de dicha posición, no podrá reclamarle en la jugada inmediata, esta regla esta para que no se entre en un bucle infinito en el que se van comiendo las fichas uno a otro.

Como podemos observar, en la segunda imagen, el jugador de Negras, come una ficha del jugador Blancas (1), entonces el jugador de blancas no puede volverle a comer esa ficha recién puesta, por eso mueve blanca en otro sitio (2).El jugador Negras, sigue con su jugada (3), entonces cuando le vuelve el turno a Blancas si puede volver a comer la ficha de Negras (4)



2.1.6. Pasar el turno

En lugar de poner una piedra, un jugador puede pasar (perder el turno). Cuando los dos jugares pasan, se acaba la partida.

2.1.7. Nivelación de partidas: Komi

Tradicionalmente el negro tenía ventaja sobre el blanco debido a que comenzaba primero, pero actualmente, esto no ocurre gracias al Komi. El Komi es muy sencillo, consiste una vez obtenido la puntuación final de la partida sumarle una cantidad de puntos determinado antes de empezar al jugador blanco. El más famoso es el de 6.5 putos

2.1.8. Nivelación de partidas: Handicap (Ventaja)

El hándicap se utiliza cuando se quiere dar ventaja a un jugador para nivelar la partida contra otro jugador de mayor rango. En ese caso el jugador de menor rango jugará con la negras y comenzará con cierto número de piedras ya colocadas sobre el tablero. Estas piezas se ponen sobre los puntos oscuros del tablero, llamados Hoshi, quedando las piezas de forma simétrica.

Otra variante es el hándicap libre, en el cual el jugador negras pone las fichas de ventaja donde él quiera.

2.1.9. Finalización de una partida

Existen diversas reglas para el conteo, siendo las más populares las japonesas:

- **Reglas japonesas:** Se suman las intersecciones que estén en control del jugador más el total de piezas capturadas.
- **Reglas chinas:** Se suman las intersecciones que estén en control del jugador y las piedras de este sobre el tablero.

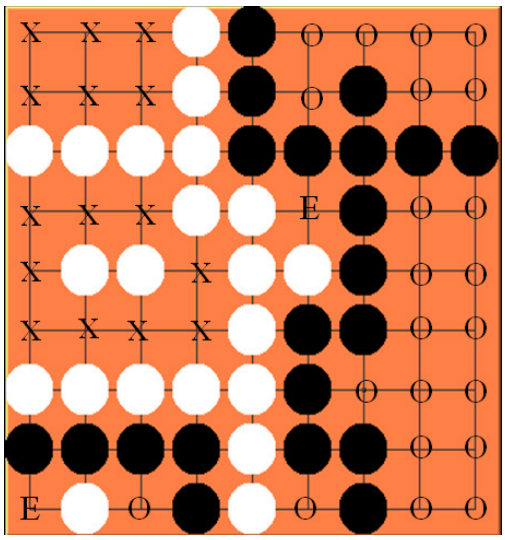
Para el presente proyecto se han utilizado las reglas japonesas.

En la siguiente imagen, a modo de ejemplo, podemos observar **las puntuaciones** si acabara así la partida:

El territorio controlado por el jugador **Blancas** está representado por las **"X"**, y son 15 los huecos controlados por él.

Por otro lado, el territorio controlado por el jugador **Negras,** está representado con **"O"**, y son 22 los huecos controlados por éste.

Finalmente, el territorio **Neutral** está representado por las **"E"** y son 2 los huecos pertenecientes a este territorio.



Capítulo 3. Algoritmo Mini-Max

3.1. Explicación del algoritmo Mini-Max

El algoritmo mini-max es un método de decisión para minimizar la pérdida máxima esperada en juegos con adversarios y con información perfecta, que se puede resumir como que vamos a elegir el mejor movimiento para mi suponiendo que el oponente va a elegir el peor para ti. Mini-max es un algoritmo recursivo, y el corte de recursión está dado por alguna de estas condiciones:

- Gana algún jugador.
- Se han explorado N capas, siendo N el límite establecido.
- Se ha agotado el tiempo de exploración.
- Se ha llegado a una situación estática donde no hay grandes cambios de un nivel a otro.

En nuestro caso, el **corte de recursión** está determinado por uno de estas dos situaciones; o bien se llega a la final de partida, o bien hemos explorado las N capas o niveles que hemos determinado.

El **espacio de estados** se representa mediante árboles alternados, donde:

- Nodo: representa una situación del juego, en nuestro caso representa la situación del tablero.
- Sucesores de un nodo: Son situaciones del juego a las que se accede por movimientos legales aplicando sus reglas, en nuestro caso, llegaremos a diferentes nodos, poniendo fichas en el tablero, siempre cuando ese movimiento sea legal.

3.2. Cómo implementar el algoritmo Mini-Max

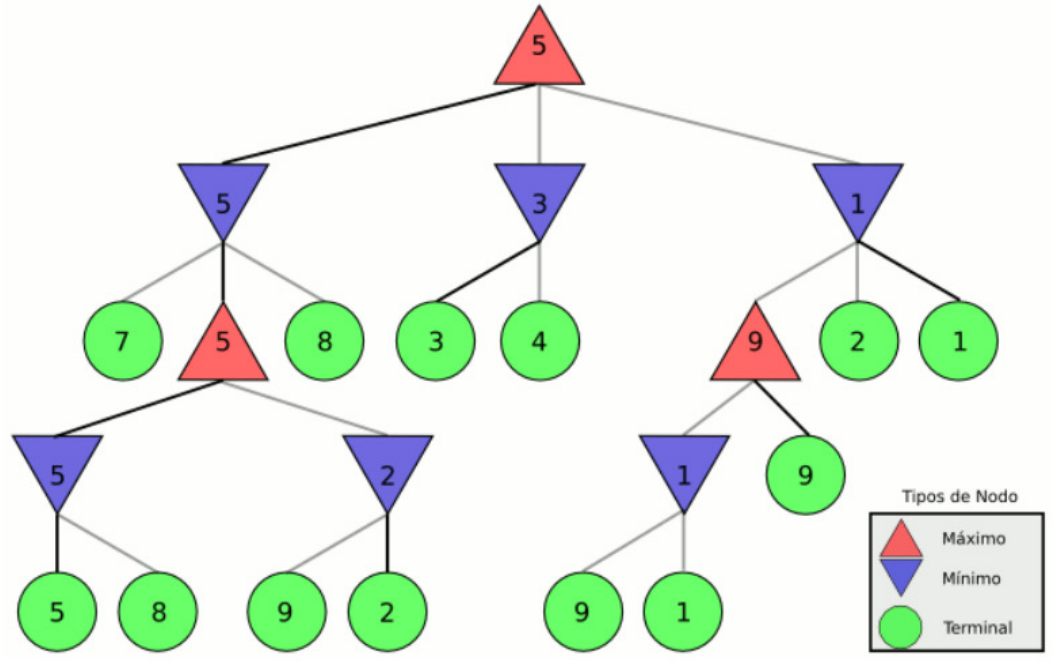
1. Generación del árbol de juego. Se generarán todos los nodos hasta llegar a un estado terminal.
2. Una vez que hemos llegado a un nodo terminal, calculamos el valor que va a tener este nodo, con la función de evaluación.

3. Calculamos el valor de los nodos superiores a partir del valor de los inferiores. Alternativamente se elegirán los valores mínimos y máximos representando los movimientos del jugador y del oponente, es decir el nodo padre de todo el árbol, será un nodo maximizador, los hijos de este padre serán todos minimizadores, y los hijos de estos hijos volverán a ser maximizadores, y así sucesivamente.

4. Al final, elegiremos la jugada valorando los valores que llegan al nivel superior.

5. La función de evaluación, definirá lo bueno que es el estado donde estamos (en nuestro caso “miraremos” el tablero), es decir, si el estado actual es beneficios para nosotros la función de evaluación le asignará un numero positivo a este estado o nodo, pero si positivo este estado para nosotros, entonces le asignará un numero negativo, es decir los números positivos indican la ventaja que tenemos nosotros, y los negativos, la ventaja que tiene el contrario.

6. El maximizador busca movimientos que le conduzcan al mayor número positivo. El minimizador busca movimientos que le conduzcan al menor número negativo.



Como podemos observar en la imagen anterior, vamos subiendo desde los nodos terminales hacia arriaba, si el nodo es minimizador cogerá el menor valor, y si es maximizador, cogerá el mayor valor, como vemos, el nodo padre del árbol a obtenido el valor heurístico (5), entonces el movimiento que haremos es el de su hijo izquierdo.

3.3. Cómo hemos implementado Mini-Max en el juego Go

Como sabemos, para implementar el algoritmo mini-max necesitamos una función recursiva, la cual la hemos implementado de la siguiente manera.

Cada vez que el jugador máquina quiera mover ficha, vamos a llamar a la función recursiva mini-max a la cual le pasaremos el tablero actual, la profundidad en la que estamos, y el turno del jugador que le toca. Si la profundidad es 0 o si llegamos al fin de la partida, entonces calcularemos el valor heurístico del nodo actual (pasándole el tablero), y si no, llamaremos a maximizador o minimizador, dependiendo de si el turno es del jugador max, o del jugador min.

La función maximizador y minimizador es prácticamente igual, solo difieren en que la primera coge la ficha para ponerla si su heurística es mayor que la máxima de las ya tratadas, y el minimizador la cogerá si su heurística es menor que ya las tratadas. A parte de lo anterior, en estas funciones se calcula todos los posibles sucesores para el estado actual, que serán todas las fichas que podemos mover, una vez que hemos listados todas esas fichas, pues llamaremos otra vez a mini-max con el nuevo tablero habiendo puesto cada vez, una de estas fichas.

3.4. Función de evaluación en el juego GO

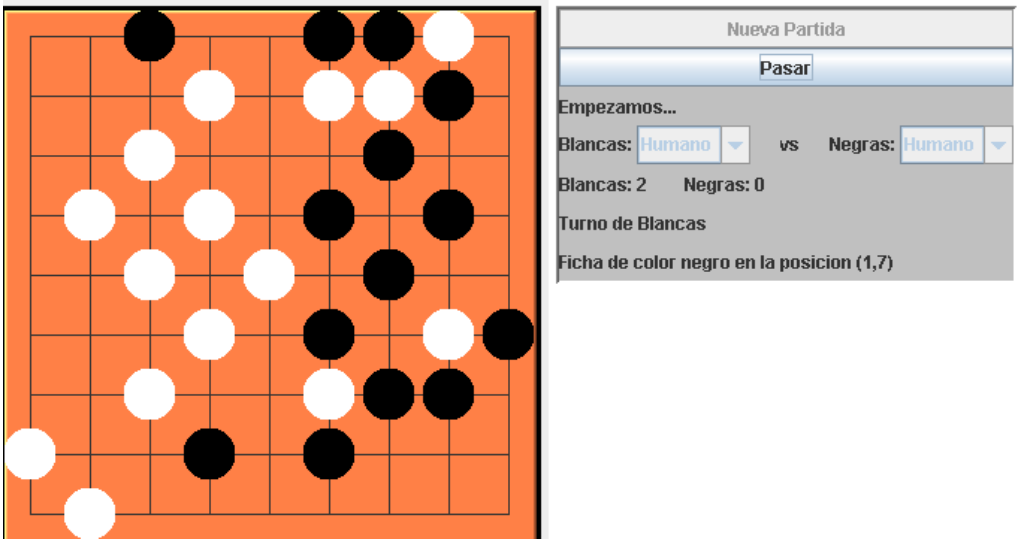
3.4.1. Función de evaluación

Este es uno de los puntos más importantes en mi proyecto.

Como hemos explicado antes, la función de evaluación es la encargada de dar un valor heurístico a cada nodo terminal. En nuestro proyecto, cada nodo está representado por la situación del tablero.

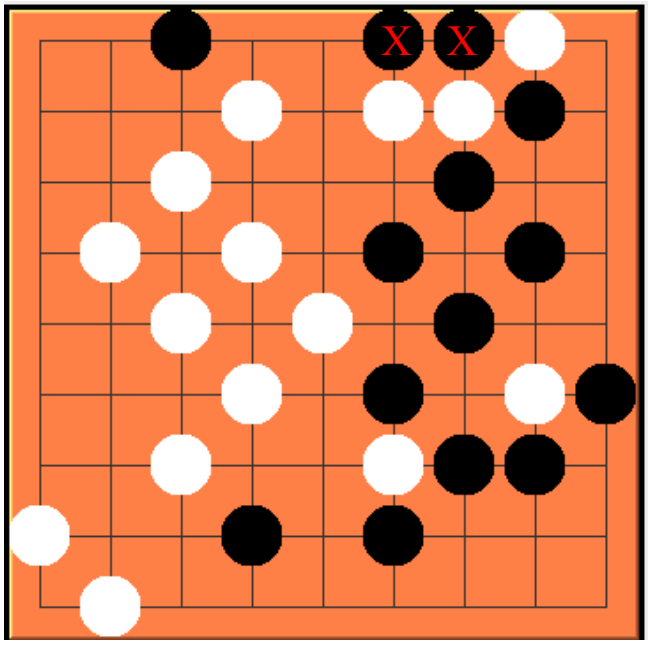
En nuestra función de evaluación vamos a hacer lo necesario para devolver un número positivo si es ventajoso para el jugador max, pero un número negativo si no es ventajoso para el mismo, para ello hemos tenido en cuenta las siguientes situaciones:

- Las fichas capturadas por cada jugador: Si el jugador max, es el jugador de las fichas Negras, entonces los puntos serían, capturadas por negras menos las capturadas por blancas por el numero heurístico que le quedemos dar a cada ficha de ventaja que le llevamos, en mi caso le he puesto un 7. Si las fichas que he comido yo es mayor que las del contrario, será un numero positivo, que lo multiplicaré por 7, en cambio si el contrario ha comida mas fichas que yo, entonces será un numero negativo el que se multiplique por 7.Lo mismo seria si el jugador max es el de las fichas Blancas, solo que será las capturadas por blancas menos las capturadas por negras.



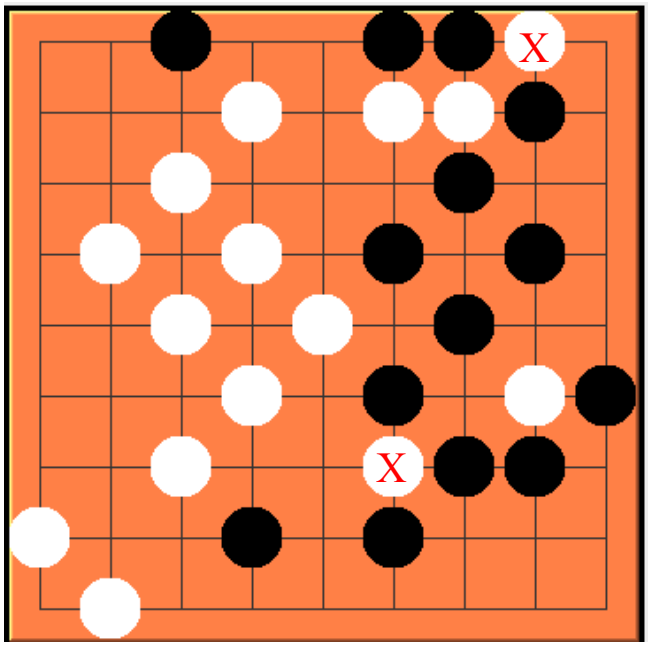
Vamos a suponer que el jugador MAX es el jugador Negras, y nos encontramos con este estado en el tablero, en un nodo terminal. La función de evaluación mirará el numero de fichas capturadas por cada jugador, en este caso el jugador Blancas a capturado 2 fichas y el jugador Negras a capturada 0.Como el jugador Max es el jugador de Negras, como es de suponer, esta heurística dará un valor negativo (si solo tenemos en cuenta las fichas comidas), ya que el jugador de Blancas o comido más fichas que el de negras. En este caso, como a las fichas comidas le hemos dado un peso de 7, pues restaremos a la heurística en este caso 14 puntos.

- Mis fichas en atari: Vamos restar 3 unidades a las variables en la que estamos guardando la heurística, le resto 3 porque tener fichas en peligro una desventaja hacia el jugador max.



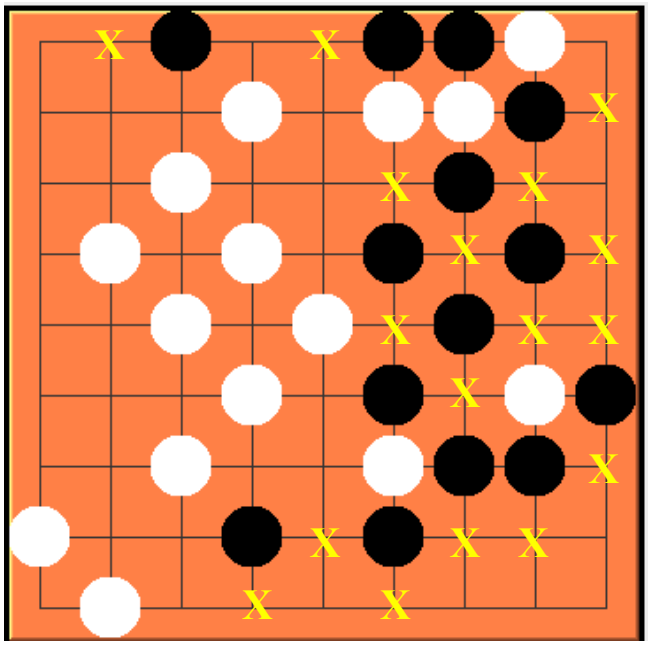
En este caso, más a recorrer todo el tablero, y contar cuantas fichas tengo en atari (peligro de ser comidas).
En la imagen de la izquierda tenemos marcadas las fichas que están en peligro. En este ejemplo son 2 las fichas que tenemos en atari, por lo que restaremos 6 puntos a la heurística, es decir 3 puntos por cada ficha en peligro.

- Sus fichas en atari: Por cada ficha que tenga el oponente en peligro vamos a sumar 3 unidades a la variable en la cual guardamos la heurística, ya que si el oponente tiene fichas en peligro es ventajoso para el jugador max.



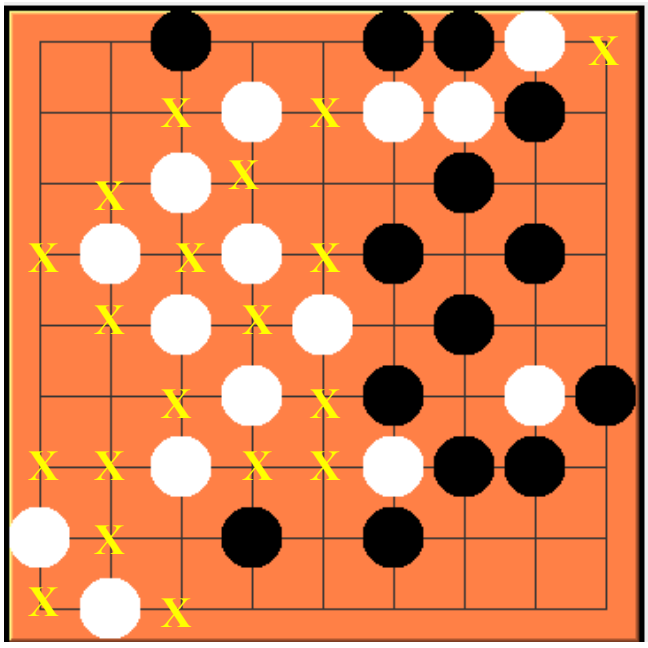
Para mirar este apartado de la función de evaluación, también a recorrer todo el tablero, y contar cuantas fichas tiene el contrincante en atari (peligro de ser comidas).
En la imagen de la izquierda tenemos marcadas las fichas que están en peligro. En este ejemplo son 2 las fichas que tenemos en atari, por lo que sumaremos 6 puntos a la heurística, es decir 3 puntos por cada ficha del contrario

- Cercos o huecos dominados por nosotros: por cada hueco que está en nuestro domino vamos a sumarle 5 puntos a la heurística, le sumamos un poco menos que por las capturadas, por que las fichas capturadas ya son puntos fijos a la hora de contar cuando se acabe la partida, pero los huecos dominados puede cambiar durante la partida, ya que una zona dominada por ti, te la puede "conquistar" el oponente.



Para hacer este apartado en la función de evaluación, hacemos un recorrido por todo el tablero, y vamos a contar los cruces controlados por cada jugador. Para cada cruce vamos a mirar sus cruces adyacentes, si el número de fichas del jugador MAX es superior que las del contrario en los cruces adyacentes, contaremos el cruce actual como controlado por el jugador MAX. En la imagen de la izquierda, tenemos marcados los cruces dominados por el Jugador Negras (suponemos que es MAX). Si el número de fichas es igual de uno y otro jugador, el cruce será neutral.
En este caso, las Negras controlan 17 cruces, como le hemos dado un peso de 5, pues sumariamos a la heurística 17*5= 85 puntos.

- Cercos o huecos dominados por el oponente: por cada hueco que está en dominio de nuestro rival le restaremos 5 a la heurística, porque contra más huecos domine el contrario, pero es para el jugador max.



En este apartado de la función de evaluación es similar al anterior pero en este caso contaremos los cruces controlados por el contrincante.
En el estado del tablero de la imagen de la izquierda, el jugador Blancas tiene 19 cruces controlados, es decir, restaremos (porque suponemos que el jugador Negras es MAX) a la heurística 19 *5 = 95 puntos.

Una vez que hemos tratado todas estas situaciones, obtendremos un valor heurístico, el cual se le asignará al nodo actual.
En los ejemplos que hemos hecho arriba, este nodo obtendrá:

-14 - 6 + 6 + 85 - 95 = - 24 puntos

Es decir, este estado terminal tendría una heurística – 24, es decir es un estado perjudicial para el jugador MAX, y por lo contrario, beneficiario par el jugador MIN.

3.4.1. Cómo hemos creado la función de evaluación en nuestro proyecto

Para crear la función de evaluación, a este método le hemos pasado a través del mini-max, el tablero, el tablero anterior (es decir el estado del tablero antes de poner la última ficha), el jugador MAX (el jugador que es max, que será o bien el jugador blanca o el jugador negras), el número de capturadas por el jugador blancas, y por el jugador negras.

Creo una variable que se llama puntos, la cual la inicializaré a 0. El primer punto a tratar en la función de evaluación, será el número de capturadas, le sumaremos a puntos, la diferencia entre mis fichas comidas y las del contrario,

multiplicadas por 7, es decir si por ejemplo yo he comido 4 fichas y el contrario me ha comido 2, entonces sumaré 14 puntos a la heurística.

El segundo punto a tratar, es las fichas que tengo en peligro. Para calcular esto, he creado un método al cual le pasamos el tablero anterior, el tablero actual, y el jugador al que quieres contar las fichas que tiene en peligro. Esta función irá recorriendo toda las posiciones del tablero, y pondrá una ficha del jugador contrario al que queremos contar, en todas las "casillas posibles" (en las únicas que son sucesoras, es decir no podrá poner un ficha si por ejemplo incumples la regla del suicidio), si alguna de estas fichas consigue capturar un número de fichas, sumaremos esas fichas a la variable fichas en peligro, y le devolveremos la suma de las fichas que tenemos en peligro. En la variable puntos (es valor heurístico), restaremos un puntos por cada ficha que tenga en peligro (restamos porque es perjudicial para el jugador max), y sumaremos si es el contrario el que tiene fichas en peligro.

Lo último que tendremos en cuenta en esta función de evaluación será el número de "cercos" controlados actualmente por nosotros (ya que estos cercos pueden ser conquistados más adelante por el contrario). Para calcular esto, hemos creado otro método, al cual le pasaremos el tablero, y el jugador que es MAX. El método hará un recorrido por todo el tablero, y mirará si esa posición está controlada por alguna de nuestras fichas, para ello mirará si alrededor suya hay más fichas mías que fichas del contrario. Por cada "casilla" controlada por nosotros, le voy a dar un número heurístico 3, le doy un número más bajo que si comeríamos ficha, ya que una ficha comida se va a contar directamente al final de la partido, pero una casilla controlada actualmente, puede ser pérdida a lo largo de la partida.

Una vez que le hemos dado el valor heurístico al estado del tablero en un nodo terminal, si es jugador MAX, se guardará el camino con mayor heurística, pero sí más adelante viene un sucesor mejor, pues cambiaremos de camino en el árbol. El jugador MIN hará lo mismo, pero para él, un valor heurístico menor al actual, será mejor y este pasará a ser el nuevo camino.

3.5. Optimización del algoritmo Mini-Max con la poda α-β

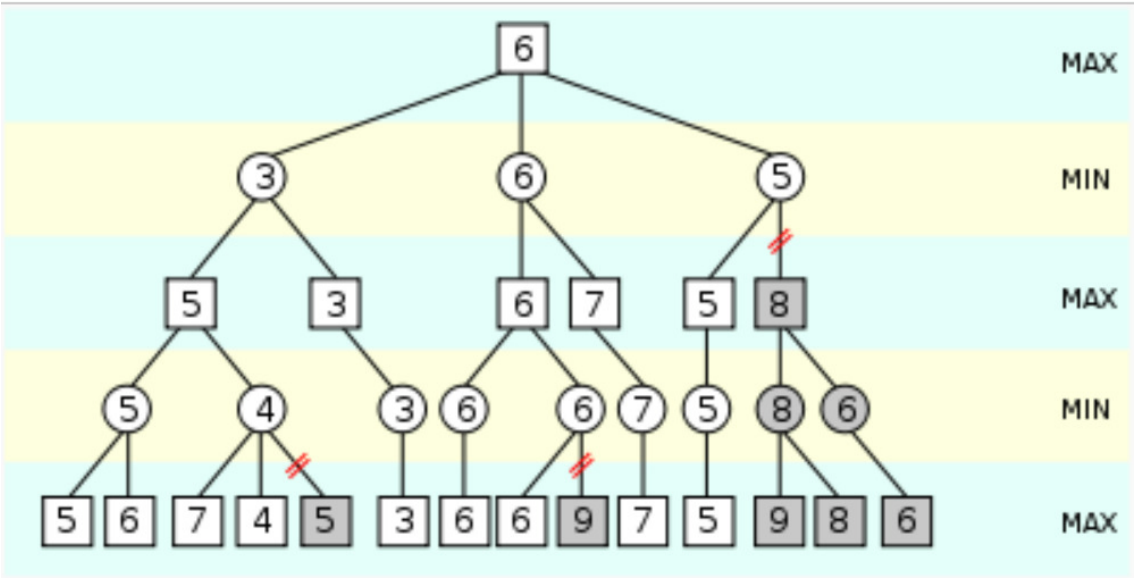
La poda alfa beta es una técnica de búsqueda que reduce el número de nodos evaluados en un árbol de juego por el algoritmo Mini-Max. Se trata de una técnica muy utilizada en programas de juegos entre adversarios como el ajedrez, el tres en raya o el Go.

El problema, de la búsqueda Mini-Max es que el número de estados a explorar es exponencial al número de movimientos. La técnica de poda alfa-beta trata de eliminar partes grandes del árbol, aplicándolo a un árbol Mini-Max estándar, de forma que devuelva el mismo movimiento, ya que lo poda de dichas ramas no influye en la decisión final.

La poda alfa-beta toma dicho nombre de la utilización de dos parámetros que describen los límites sobre las valores hacia atrás que aparecen a lo largo de cada camino.

- **α** es el valor de la mejor opción hasta el momento a lo largo del camino para MAX, esto implicará por lo tanto la elección del valor más alto.
- **β** es el valor de la mejor opción hasta el momento a lo largo del camino para MIN, esto implicará por lo tanto la elección del valor más bajo.

Esta búsqueda alfa-beta (α- β) va actualizando el valor de los parámetros según recorre el árbol. El método realizará la poda de las ramas restantes cuando el valor actual que se está examinando sea peor que el valor actual de alfa o beta para MAX o MIN respectivamente.



En la imagen de arriba, podemos observar cómo se aplica la poda al árbol. El padre de los nodos hojas más a la izquierda, etiquetados con 5 y 6 respectivamente, deberá escoger un valor para beta que será el menor de sus hijos, en este caso la beta será 5. Si seguimos expandiendo el árbol, el nodo padre del nodo min que acabamos de tratar, cogerá el valor máximo de sus hijos para alfa, en este caso como solo hay un valor, alfa sería 5. Si continuamos expandiendo el árbol hacia sus hijos, llegamos al nodo con valor heurístico igual a 7, como alfa (5) sigue siendo menor que beta (7) se sigue expandiendo el árbol, y nos encontramos con el nodo con valor heurístico (4), en este caso beta será ahora 4 porque es menor que 7, y ahora sí alfa es mayor que beta, entonces se hace la poda, y no se evaluaría el nodo con valor 5. Esto se iría haciendo por todo el árbol.

La eficacia de la poda alfa-beta depende del orden en el que se examinan los sucesores, es decir, el algoritmo se comportará de forma más eficiente si examinamos primero los sucesores que probablemente serán los mejores.

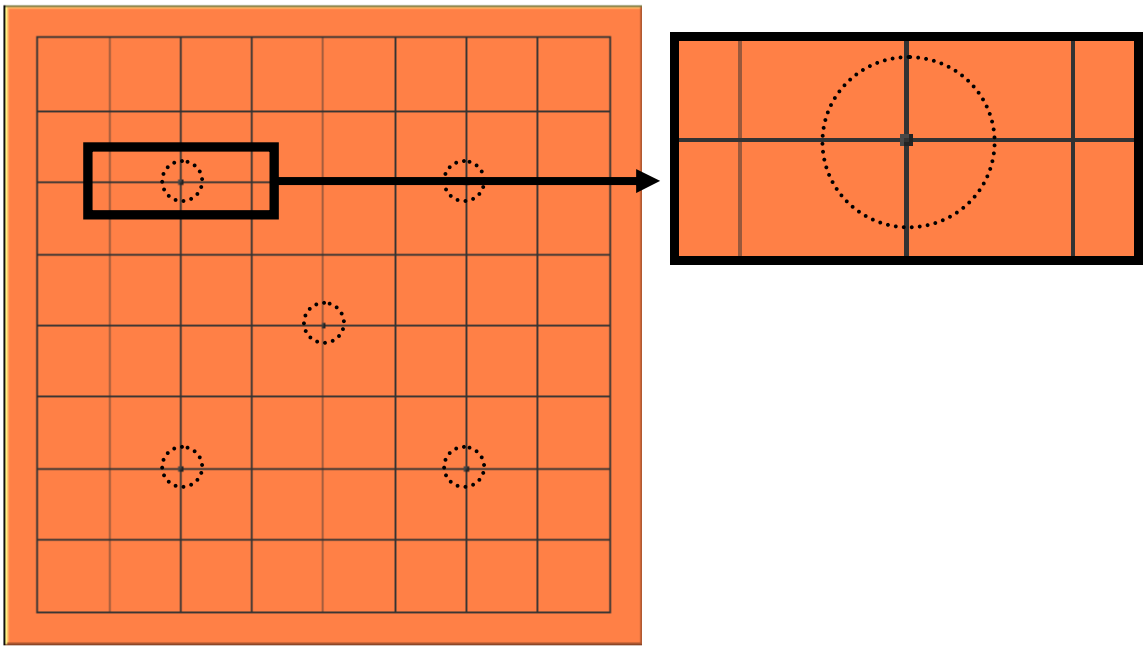
3.6. Aplicación de una jugada de apertura

En primer lugar, sería conveniente comentar que en este tipo de juegos, basados en técnicas relacionadas con las heurísticas, es muy frecuente el uso de técnicas de estrategia al comenzar la partida.

En el caso del Go, el jugador Máquina, antes de utilizar el mini-max con la poda alfa-beta, utiliza una **jugada de apertura** en sus 2 ó 3 primeros movimientos, donde ocupará los **sitios estratégicos** (marcados en la siguiente imagen mediante los círculos con línea discontinua), para ir marcando el territorio que quiere conquistar. Para ello hemos utilizado una función que consiste en detectar si un sitio estratégico o sus alrededores han sido ocupados, ya que de ser así ocuparemos otro de los sitios estratégicos que se encuentre vacío.

Además, en el tercer movimiento, si la casilla del centro y sus casillas de alrededor no están ocupadas, se ocupará ésta (la casilla central también pertenece al conjunto de “sitios estratégicos”) perteneciendo así el movimiento a la jugada de apertura. En cambio si se encuentran ocupadas pasaremos a aplicar directamente el método del mini-max con la mejora de la poda alfa-beta.

Esta **técnica estratégica** además de hacer mejorar la heurística, también nos ayudará a que en los primeros movimientos, en los que en condiciones normales el mini-max recorrería muchos nodos, no sea necesario explorarlos, optimizando así el rendimiento y ganando en eficiencia.



Capítulo 4. Resultado Obtenidos

4.1. Mejoras obtenidas en el proyecto con la poda α-β

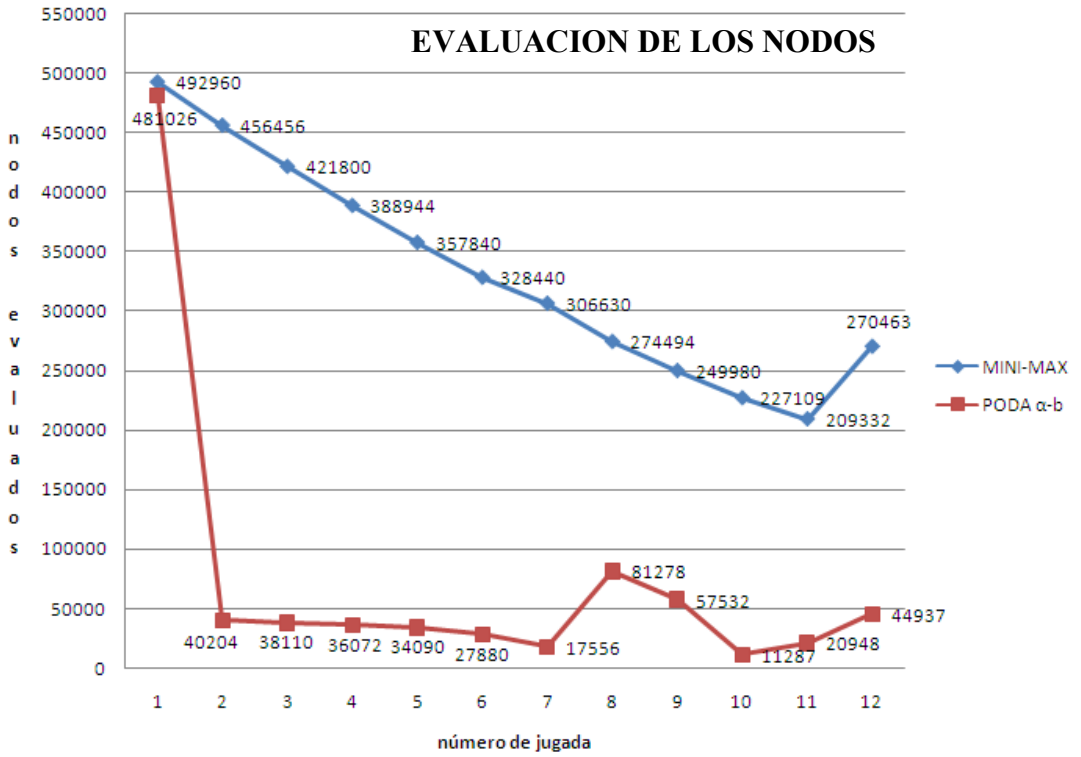
La mejoría que hemos obtenido en el algoritmo mini-max con la poda α-β es significativa, sobre todo, cuanto mayor sea la profundidad de evaluación.

Las siguientes gráficas reflejan la diferencia de número de nodos evaluados entre el algoritmo mini-max, y mini-max con la poda α-β, para las 12 primeras jugadas en una partida simulada. Para hacer las gráficas hemos jugado contra el algoritmo mini-max, y después con la mejora α-β, siempre haciendo la misma jugada, asegurándonos así que los movimientos son los mismos para uno y otro.

Nos encontramos diferentes gráficas dependiendo de la profundidad que hemos establecido, y de si hemos aplicado o no, la jugada de apertura.

4.1.1 Evaluación de nodos y del coste en tiempo con profundidad 3 sin jugada de apertura

La siguiente gráfica refleja el número de nodos evaluados para el algoritmo mini-max, y para el mini-max con la poda α-β, para las 12 primeras jugadas en una partida simulada con profundidad 3 y sin hacer la jugada de apertura.

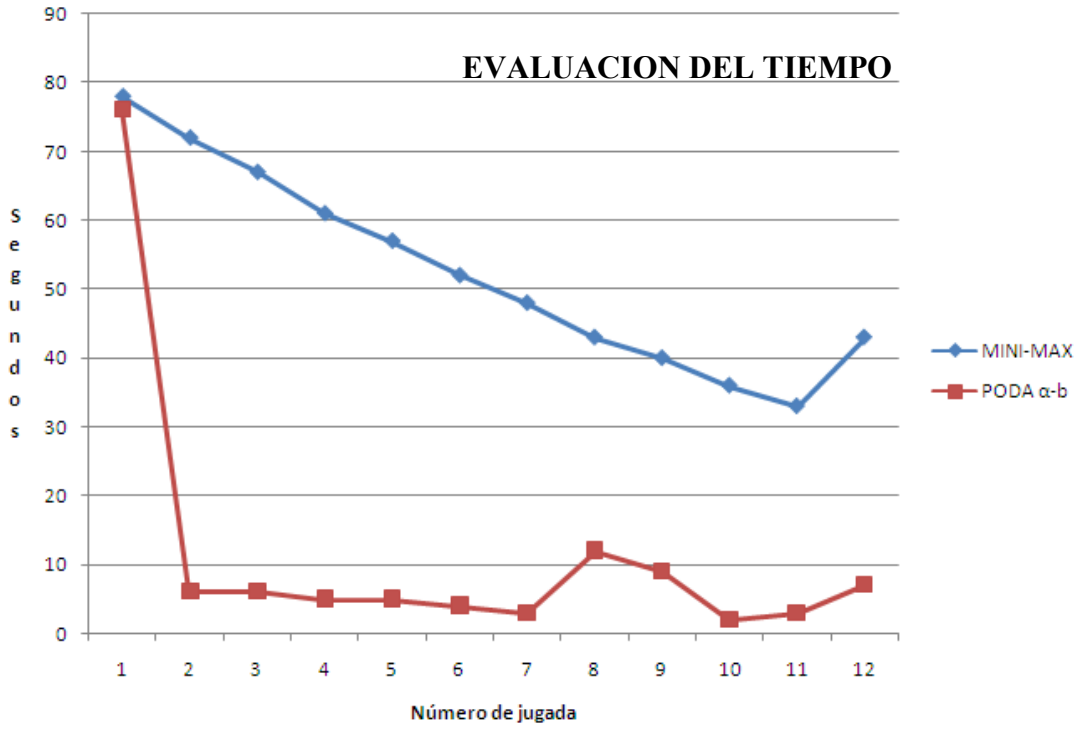


Como podemos observar en la gráfica, para el primer movimiento, el número de nodos evaluados para uno y otro es casi similar, ya que al ser profundidad 3, la heurística aún no ha tenido casi efecto, porque si recordamos, en la función de evaluación de la heurística tenemos en cuenta el número de fichas capturadas, el número de fichas en peligro, y el número de cercos controlados, y es muy difícil que en las tres primeras jugadas (profundidad 3) tengamos un cerco controlado, o que comamos un ficha, o que tengamos una ficha en peligro, es por eso, que en el primer movimiento la diferencia entre un algoritmo y otro sea tan poco significativa, ya que la poda casi no ha tenido efecto.

Si nos fijamos en el segundo y en los siguientes movimientos, podemos observar que la mejora que hemos obtenido es bastante significativa, ya que la poda ha "entrado" en juego. En el segundo movimiento el algoritmo mini-max ha evaluado 456.456 nodos, y el algoritmo con la poda alfa-beta 40.204, es decir una diferencia de 416.252 nodos.

Si sumamos los nodos totales de uno y otro al cabo de los 12 movimientos, la diferencia es abultada, el mini-max habría evaluado 3.978.448 nodos, frente a los 890.920 de la poda alfa-beta. Los nodos totales que ha evaluado el algoritmo con la poda en los 12 primeros movimientos, ya es sobrepasado por los nodos evaluados por el mini-max en los dos primeros movimientos.

La siguiente gráfica reflejará el tiempo que cuesta en mover una jugada entre los dos algoritmos, con profundidad 3 sin aplicar la jugada de apertura.



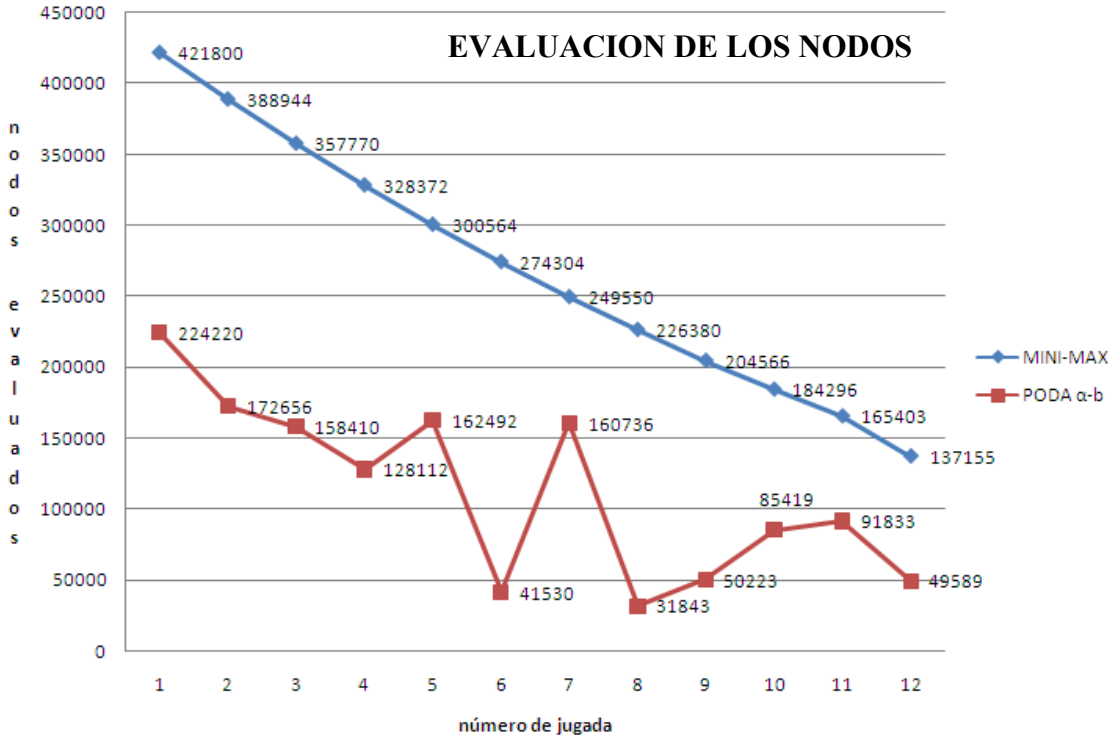
Si sumamos todos los segundos que tardan en hacer los movimientos, obtenemos los siguientes datos:

Segundos totales MINI-MAX = 630s = 10m 30s
Segundos totales PODA α-β = 138s = 2m 18s

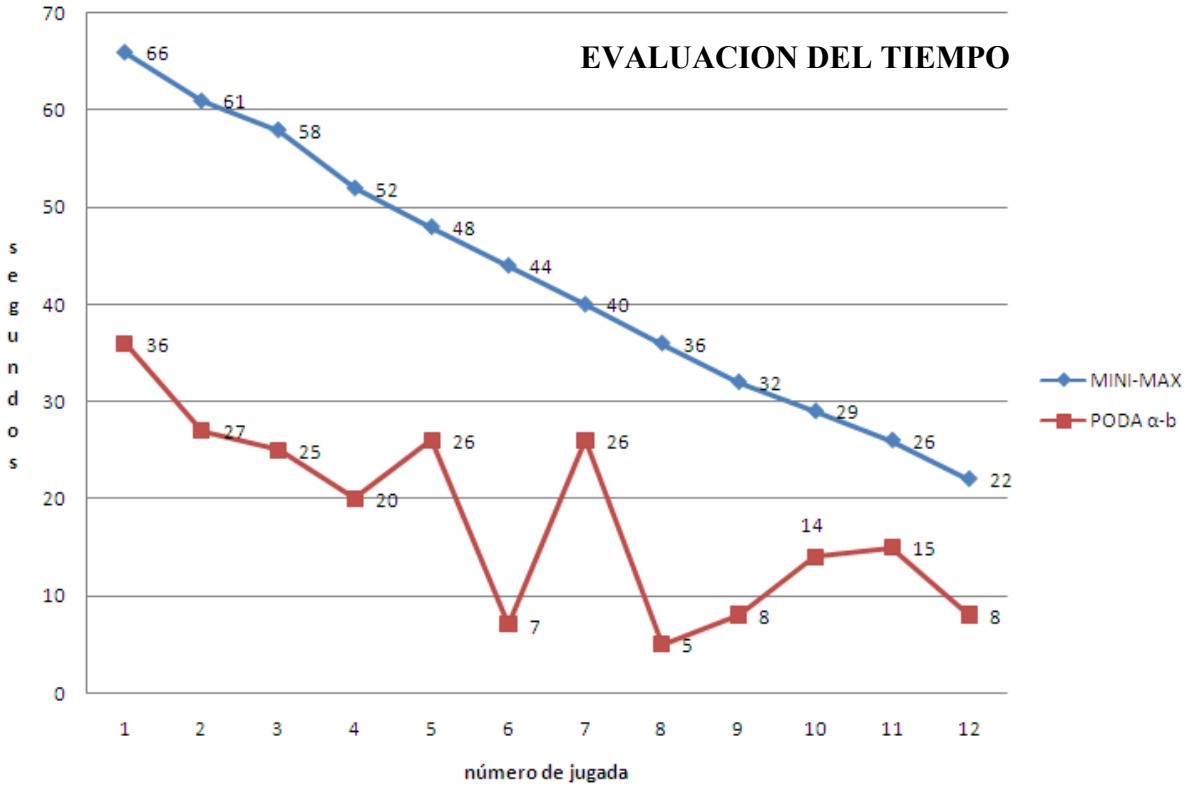
Si nos fijamos nodo a nodo, con profundidad 3 pues la diferencia no es muy alta, pero si sumamos todos, podemos observar que se ahorra mas de 8 minutos.

4.1.2 Evaluación de nodos con profundidad 3 con jugada de apertura

En esta otra gráfica, como hemos comenzado la partida con la jugada de apertura, podemos observar, que desde el primer movimiento la poda nos obtiene resultados importantes ya que en el primer movimiento, nos poda casi la mitad de los nodos. Es decir, empezar con la jugada de apertura, además de darnos resultados a la hora de la estrategia, también nos da resultados a la hora de la evaluación de nodos. Para el algoritmo mini-max, la mejora no es muy alta, pero para el algoritmo con la poda, es bastante bueno, ya que pasamos de analizar en el primer movimiento, sin la jugada de apertura de 481.026 a 224.220 con la jugada de apertura, es decir podamos más de la mitad. Si hablamos del global, pasaríamos de analizar 1.357.063 a 890.920 nodos.



La siguiente gráfica reflejará el tiempo que cuesta en mover una jugada entre los dos algoritmos, con profundidad 3 y aplicando la jugada de apertura.



Si sumamos todos los segundos que tardan en hacer los movimientos, obtenemos los siguientes datos:

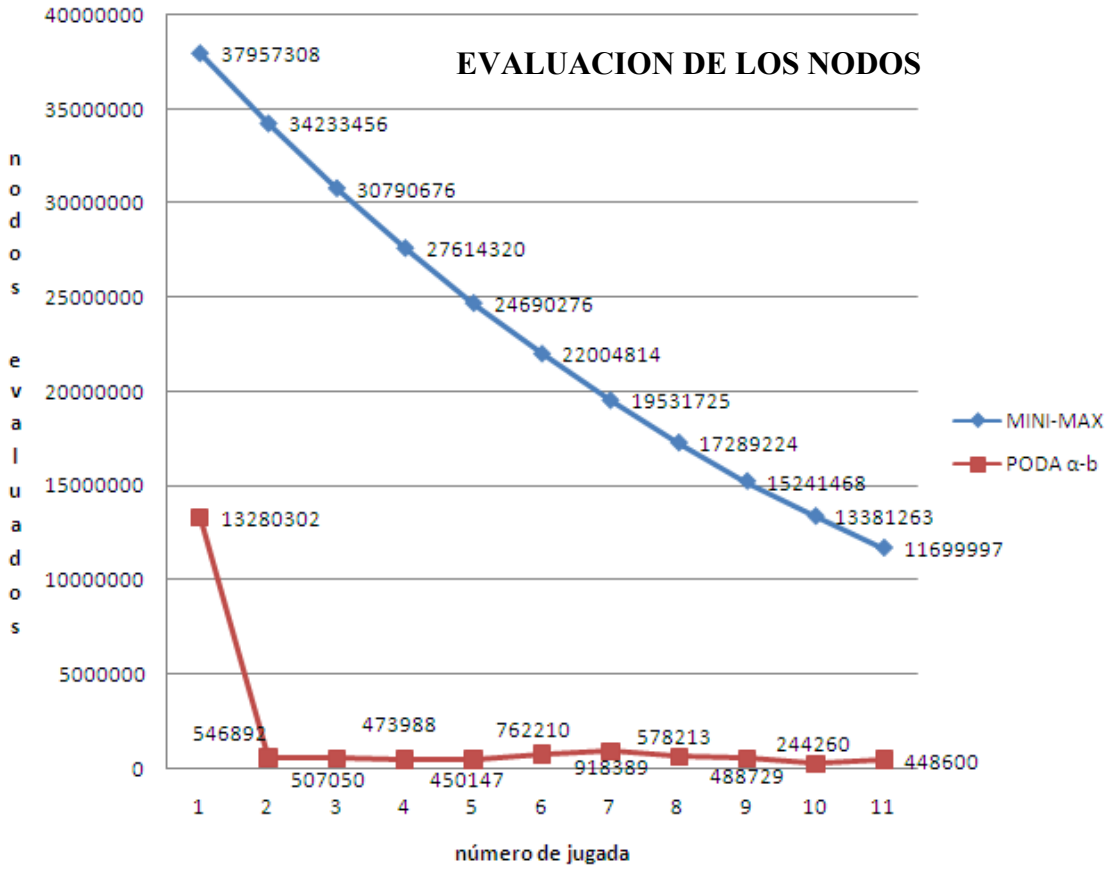
Segundos totales MINI-MAX = 514s = 8m 34s
Segundos totales PODA α-β = 217s = 3m 37s

Podemos observar que ya hay una pequeña diferencia si miramos nodo a nodo, porque en realidad la máxima mejora que obtenemos nodo a nodo es de unos 40 segundos. Aunque si nos fijamos en el coste total, podemos observar una mejora de unos 7 minutos.

4.1.3 Evaluación de nodos con profundidad 4 sin jugada de apertura

En la siguiente gráfica, en comparación con las anteriores, podemos observar la gran diferencia de nodos evaluados, debido a que el algoritmo mini-max crece exponencialmente a medida que la profundidad aumenta.

Con esta profundidad, encontramos una gran diferencia de nodos evaluados entre los dos algoritmos. La mejora que se obtiene es inmensa, pasamos de evaluar 254.434.527 con minimax, a 18.638.780 con la poda.



La siguiente gráfica reflejará el tiempo que cuesta en mover una jugada entre los dos algoritmos, con profundidad 4 sin aplicar la jugada de apertura.

Si sumamos todos los segundos que tardan en hacer los movimientos, obtenemos los siguientes datos:

Segundos totales MINI-MAX = 40384s = 673m 4s = 11h 13m 4s

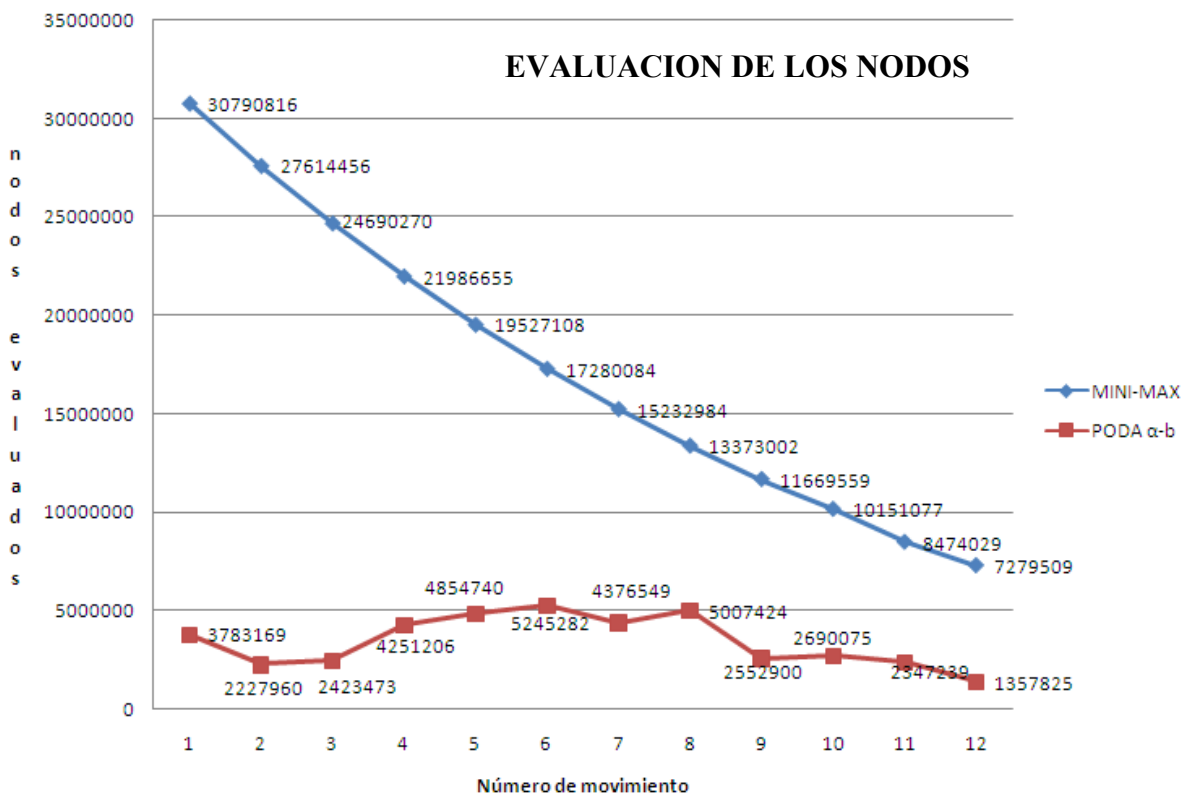
Segundos totales PODA α-β = 2968s = 49m 28s

Podemos observar que la diferencia ya es descomunal, no es lo mismo esperar una media de 2 minutos para hacer un movimiento, que esperar 58 minutos.

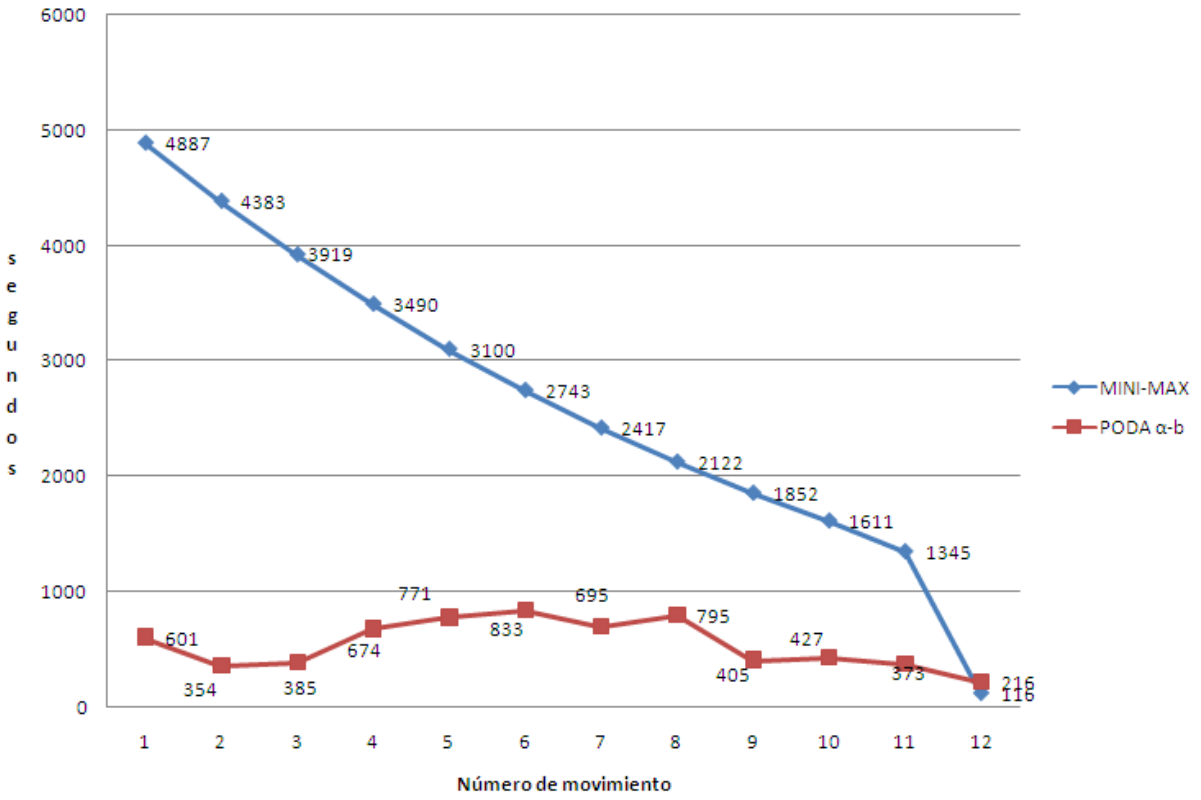
Si nos fijamos en el total, nos ahorramos más de 10 horas.

4.1.4 Evaluación de nodos con profundidad 4 con jugada de apertura

Esta última gráfica, refleja lo anteriormente explicado, es decir, que con la jugada de apertura obtenemos una gran mejoría en los dos algoritmos a la vez que nos expandimos estratégicamente en el tablero.



La siguiente gráfica reflejará el tiempo que cuesta en mover una jugada entre los dos algoritmos, con profundidad 4 y aplicando la jugada de apertura.



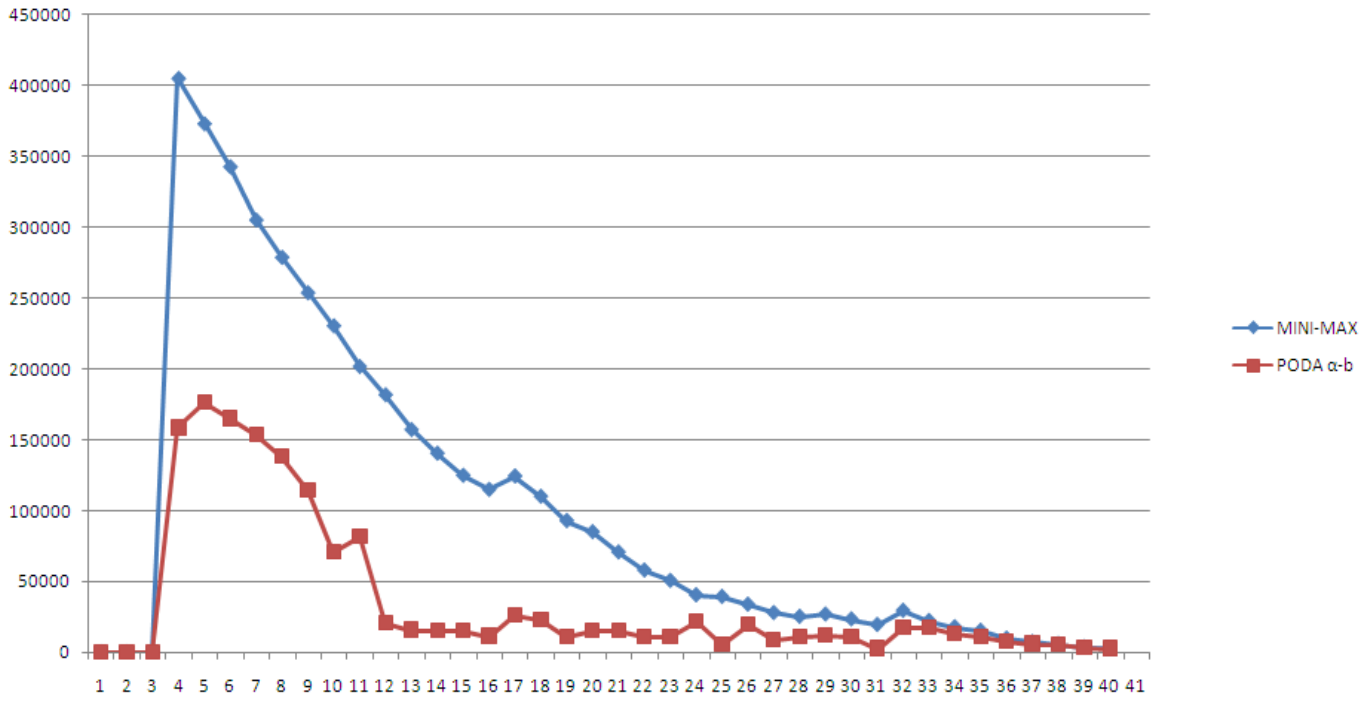
Si sumamos todos los segundos que tardan en hacer los movimientos, obtenemos los siguientes datos:

Segundos totales MINI-MAX = 31985s = 533m 5s = 8h 53m 5s
Segundos totales PODA α-β = 6529s = 108m 49s = 1h 48 m 49s

La mejoría que obtenemos aquí es tambien muy abultada, cerca de las 7horas.

4.1.5 Una partida completa de profundidad 3 con jugada de apertura

La gráfica siguiente, nos va a dar una idea de la diferencia de nodos que se evalúan entre los dos algoritmo.

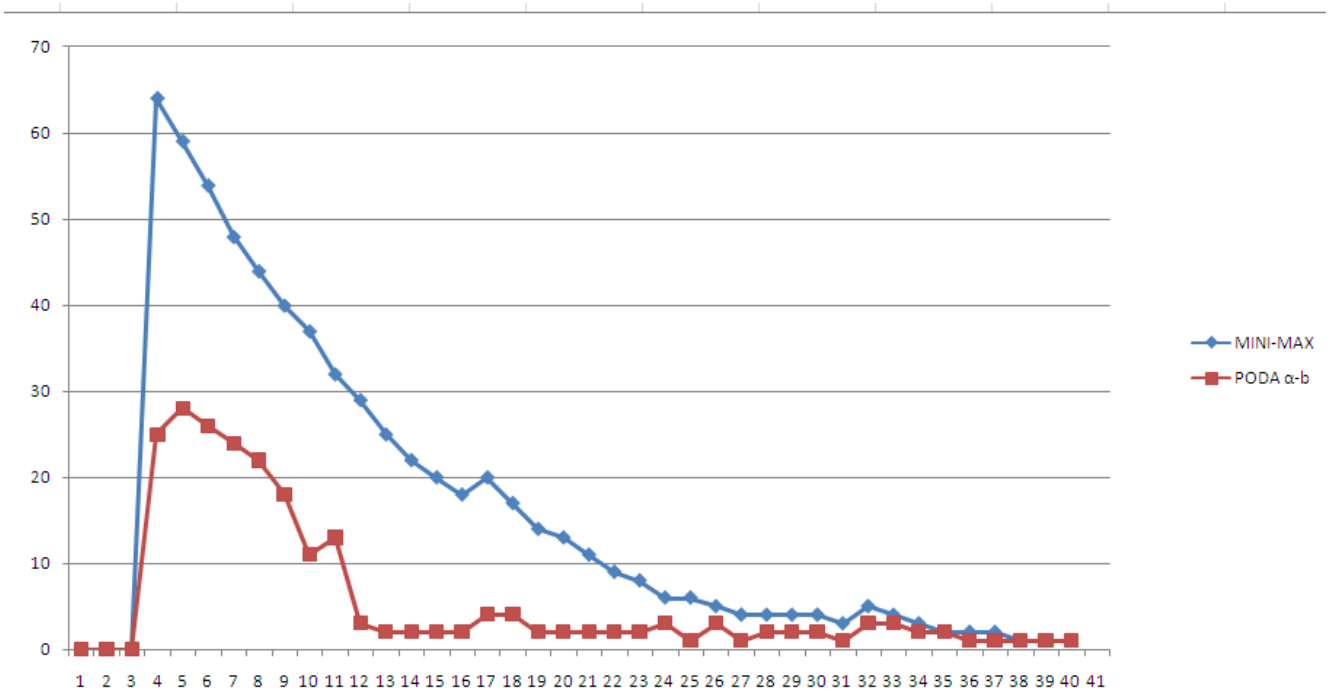


En la gráfica podemos observar la diferencia que hay durante toda una partida con profundidad 3. Podemos destacar que al principio de la partida, es cuando más efecto tiene la poda, y contra más fichas vamos poniendo en el tablero, como es lógico, menos sucesores tendremos que evaluar, y la va podando menos en proporción.

Al final de la partida, cuando quedan unos 8 movimientos, podemos observar que la diferencia de nodos que evalúan un algoritmo y otro, son casi similares.

Ahora se mostrará la misma gráfica, pero representando los segundos que tardan en hacer un movimiento.

En esta gráfica se muestra cuantos segundos tarda en hacer una jugada durante una partida completa simulada con profundidad 3.



Si sumamos todos los segundos que tardan en hacer los movimientos, obtenemos los siguientes datos:

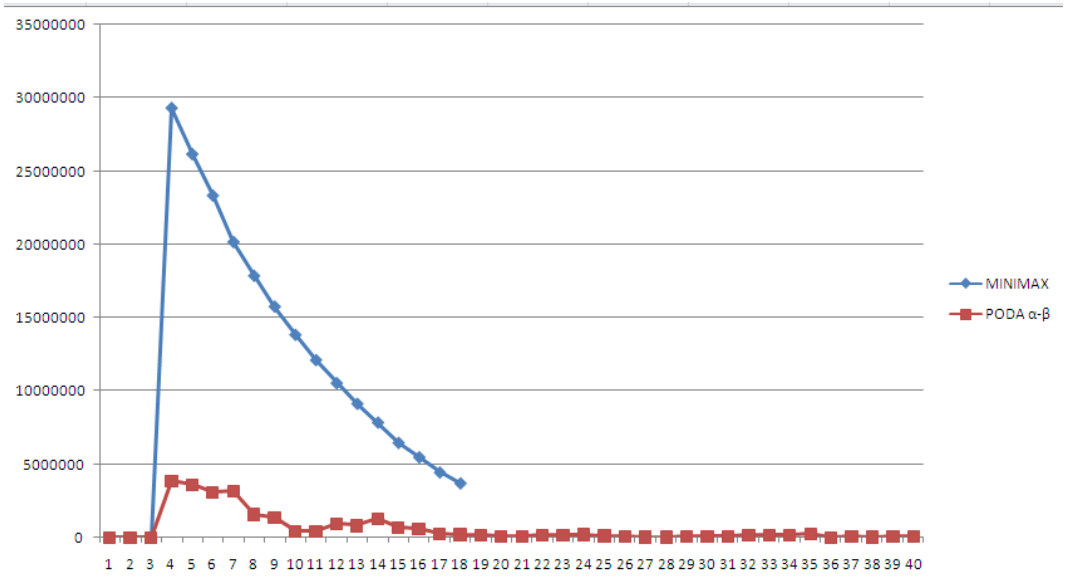
Segundos totales MINI-MAX = 641s = 10m 41s
Segundos totales PODA α-β = 226s = 3m 46s

En la gráfica observamos que con profundidad 3 conseguimos reducir el coste no en gran escala, pero suficiente para esta profundidad. Como siempre son en los primeros movimientos donde más recorte de tiempo obtenemos.

En el global de la partida, conseguimos recortar el coste de tiempo unos 7 minutos.

Como hemos observado en anteriores gráficas, con profundidad 4, la mejora de tiempo que se consigue es muy significativa.

4.1.6 Una partida completa de profundidad 4 con jugada de apertura



Esta gráfica nos da una idea de la cantidad de nodos que evalúan los dos algoritmos, y la gran diferencia entre los dos. Los nodos totales que ha evaluado con la poda son 24.579.863 de nodos, y los nodos evaluados por el mini-max hasta la jugada 18 son 205.464.554, es decir una inmensa diferencia

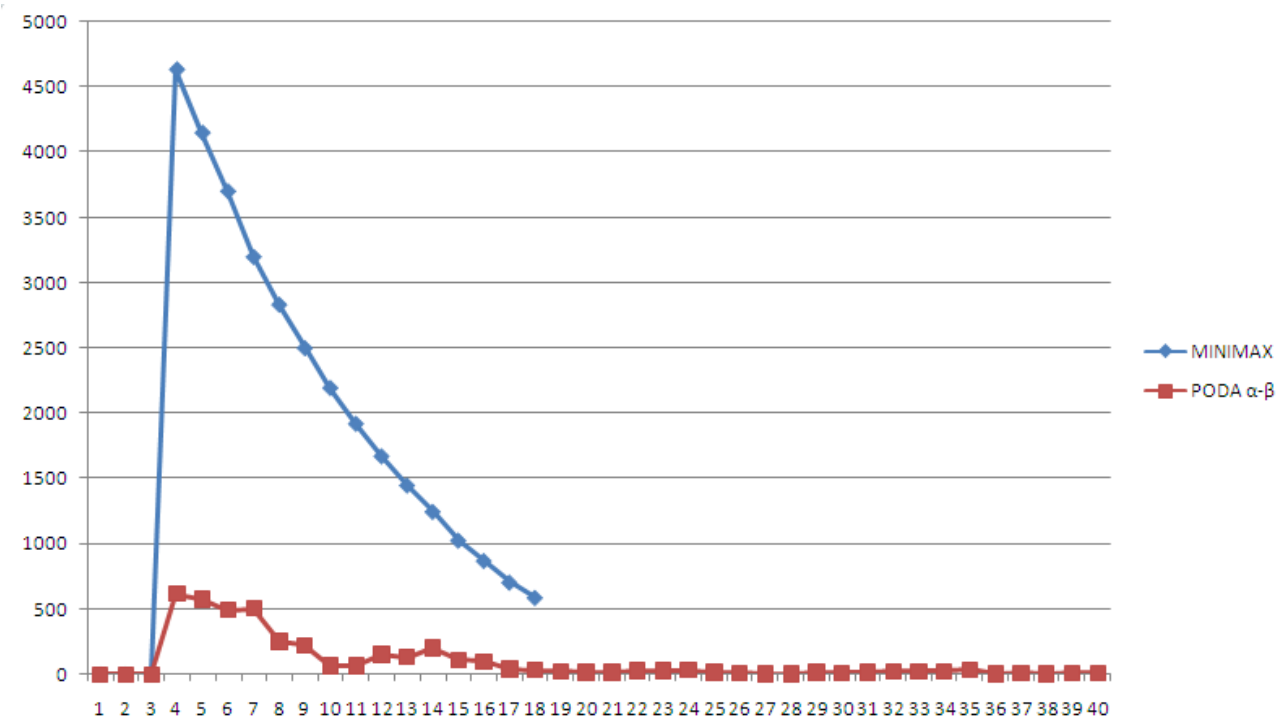
No he representado todo el algoritmo mini-max, porque el coste temporal es muy elevado.

Si sumamos todos los segundos que tardan en hacer los movimientos, obtenemos los siguientes datos:

Segundos totales MINI-MAX = 32613s = 543m 33s = 9h 3m 33s
Segundos totales PODA α-β = 3902s = 65m 2s = 1h 5m 2s

Es decir la deferencia de coste temporal es descomunal, con la poda alfa-beta se puede jugar una partida al go, pero con el mini-max necesitarías muchas horas, es decir no es muy accesible para el usuario.

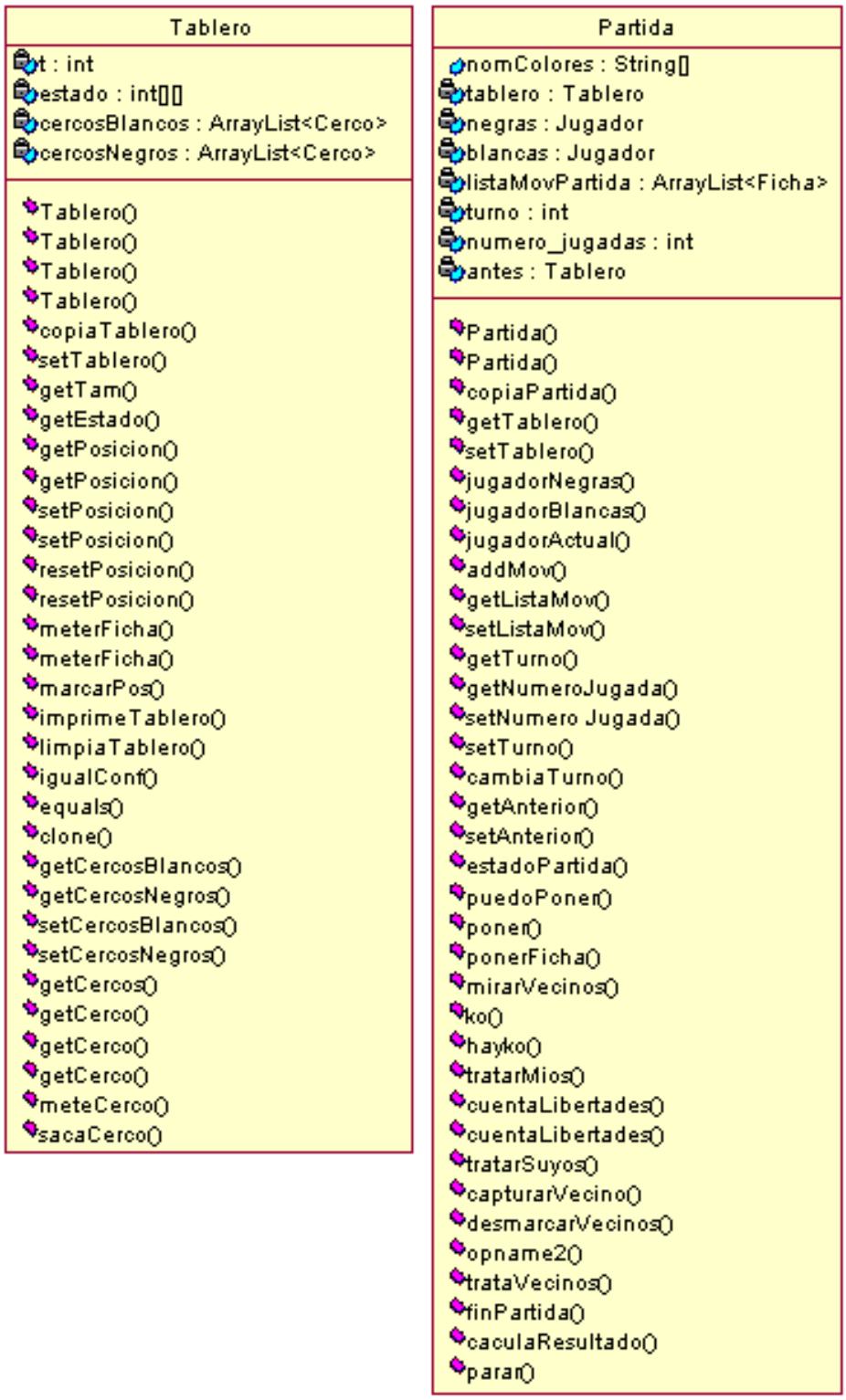
En esta gráfica se muestra cuantos segundos tarda en hacer una jugada durante una partida completa simulada con profundidad 4.

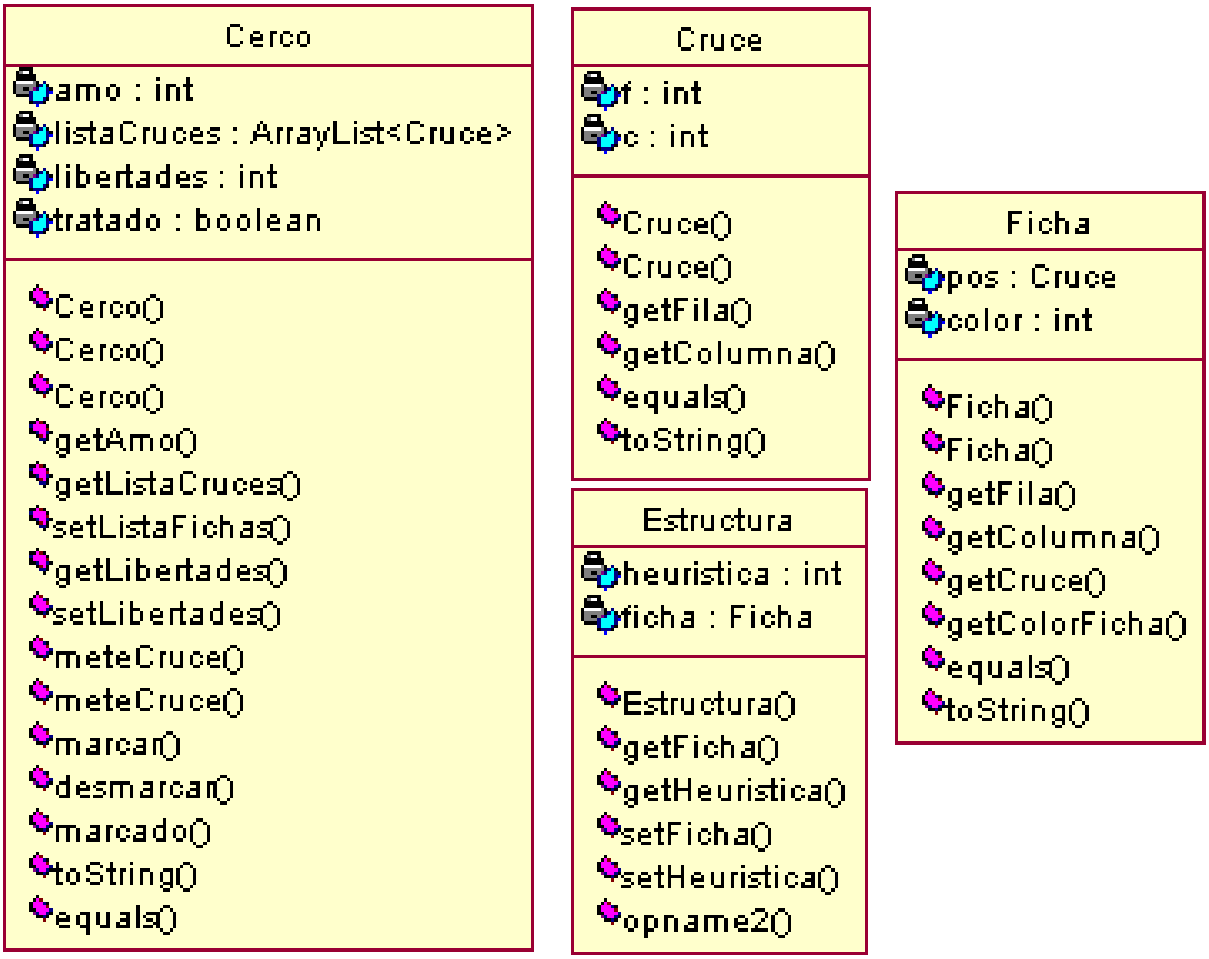


.

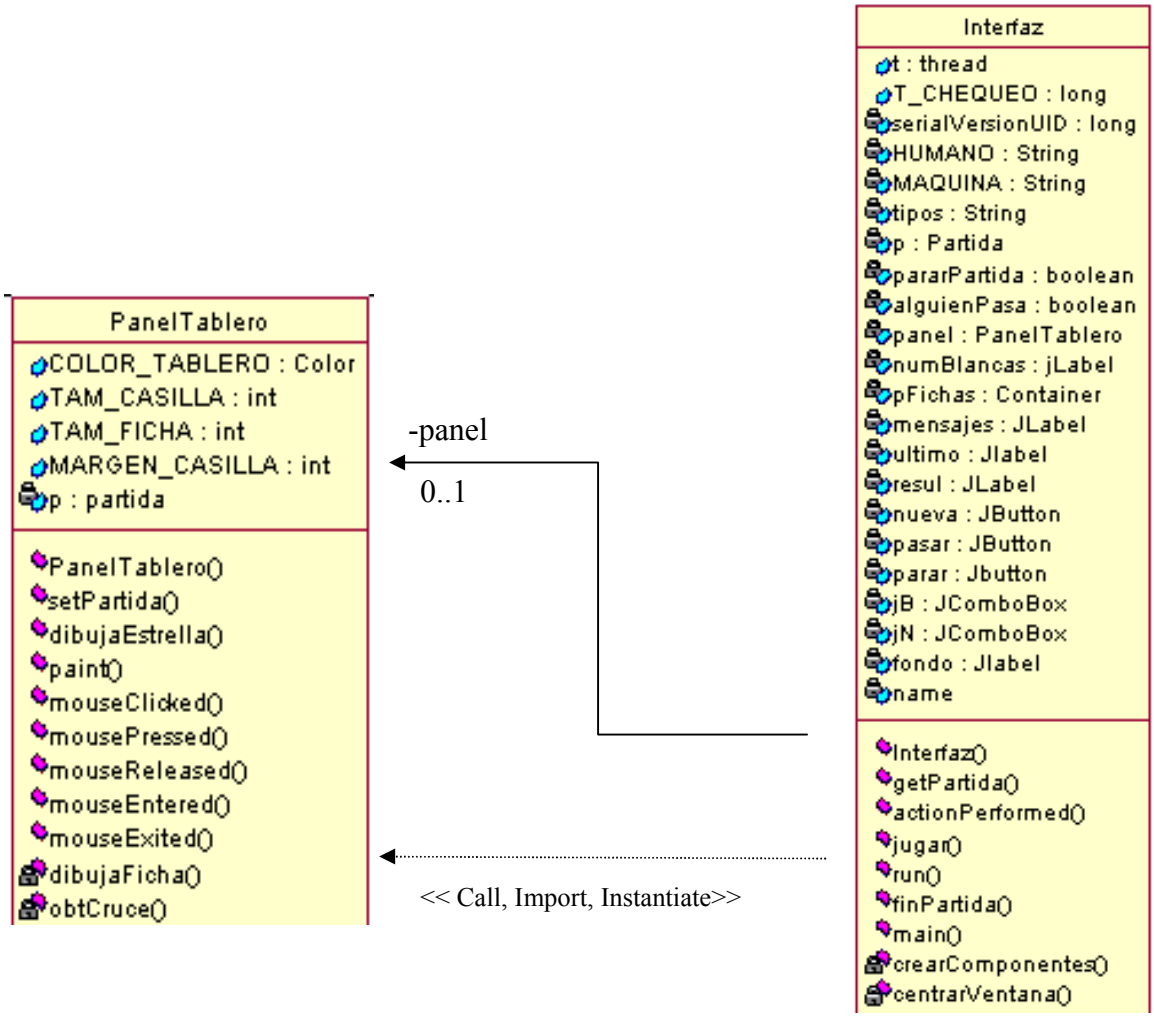
Capítulo 5. Diagrama de clases

Aquí muestro el diagrama de clases que controlan el desarrollo de la partida.

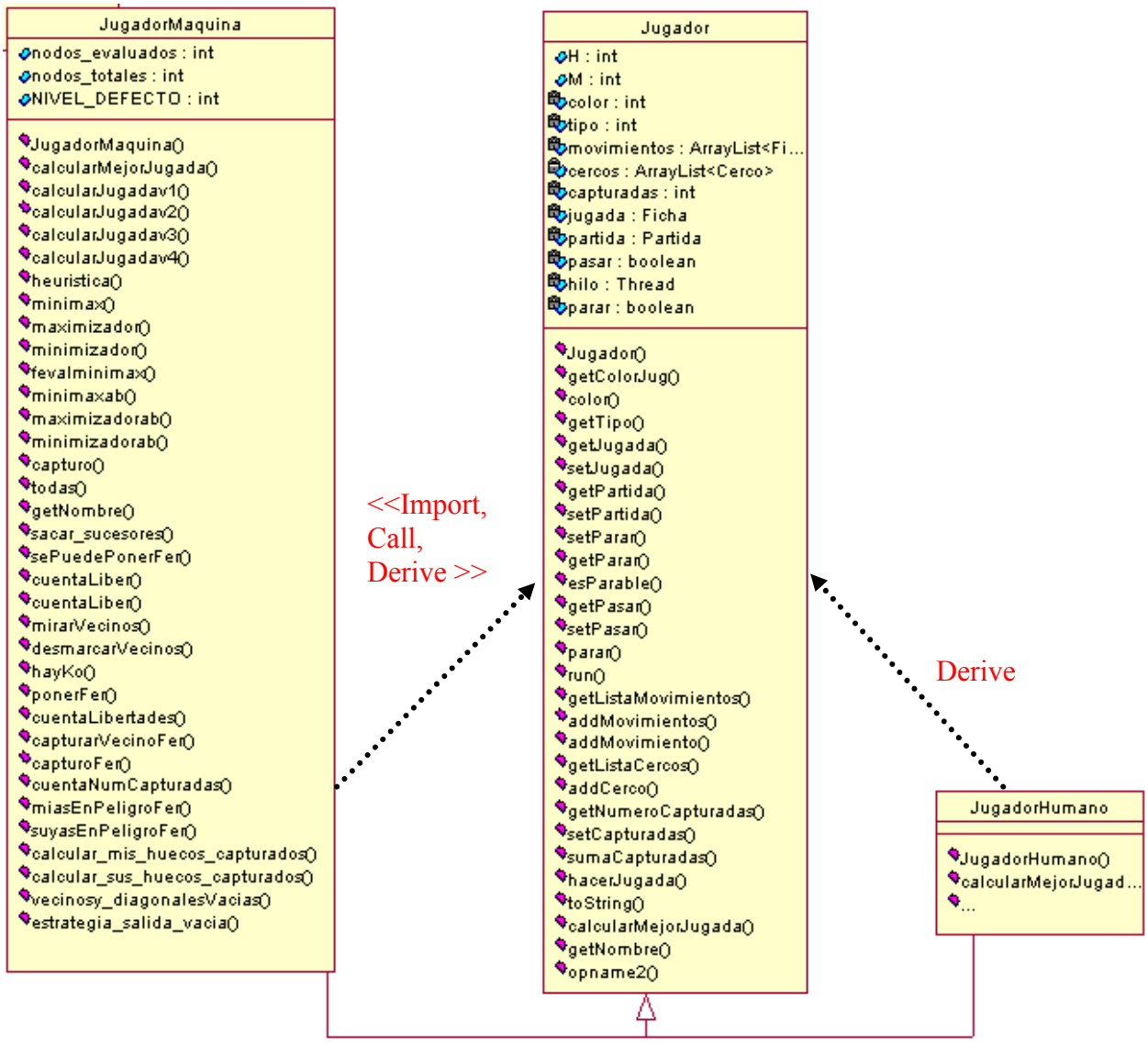




En las siguientes imágenes podemos ver las clases encargadas del GUI:



Por último, vamos a ver las clases que implementan a los jugadores, tanto el jugador máquina como el jugador Humano.



Capítulo 6. Conclusiones

El juego “go” es un juego sencillo de entender, porque las pocas reglas que tiene, son muy fáciles de aplicar en el juego. El que sea un juego, en el cual no es complicado jugar, no significa que sea sencillo saber cual son los movimientos adecuados para llevarnos a la victoria, sino todo lo contrario.

Para entrar un poco en contacto con el juego y con el jugador Máquina, implementamos dos métodos para este juego, en el primer método, entre todos sus posibles sucesores, se elige uno aleatoriamente. En la segunda función la hemos implementado de tal manera que si podemos comer ficha al contrario, lo vamos a hacer, y si no se puede, vamos a hacer un movimiento aleatorio entre los posibles movimientos.

Implementar el algoritmo mini-max a este juego no ha sido demasiado complicado, porque hay una gran información sobre este algoritmo, ya que es uno de los más utilizados para los juegos de estrategia de uno contra uno. Aunque no nos ha sido difícil implementarlo, si hemos tenido algunos problemas, debido a que algunos métodos no estaban implementados correctamente, o bien, algunas reglas del juego no estaban introducidas en él.

Incorporar la poda correctamente al algoritmo mini-max, no se necesita un gran cambio en el código, pero hay que saber muy bien que elementos incorporar. Para conseguir que lo poda funcionara correctamente, nos costó bastante trabajo, ya que no encontraba el error por el cual me funcionaba mal. Al principio no me hacia bien la poda, porque me analizaba los mismos nodos en el mini-max. Un poco más adelante si me hacia lo poda, pero me elegía mal la ficha a mover, ya que esto lo sabía, porque el mini-max con poda o sin poda tiene que hacer el mismo movimiento para un mismo estado en el tablero.

Una vez que conseguimos que funcionara correctamente el algoritmo mini-max con la poda alfa-beta, nuestro siguiente objetivo era el de crear una función de evaluación, la cual dificulte al máximo ganar al jugador máquina, es uno de los principales y más difíciles objetivos. Para obtener función de evaluación buena para este juego, es muy complicado, sobre todo porque las posibilidades de poner ficha son muy elevadas (apuntar que jugamos en un tablero de 9*9, y los profesionales son de 19*19), y el orden de la función mini-max es exponencial a la profundidad. Debido a lo descrito anteriormente, las pruebas que hemos realizado, han sido como mucho con profundidad 5, siendo casi imposible jugar en esta ultima profundidad, debido que mover una ficha en las primeras jugadas con esta profundidad hablamos de evaluar aproximadamente 123.886.725, y en tiempo estamos hablando más de 5 horas y media. Debido a esto, he decidido que la profundidad adecuada para este juego es la profundidad 4.

Para conseguir una buena función de evaluación hemos tenido en cuenta sobre todo, la forma de puntuación, que es el número de fichas comidas, y el número de cercos capturados, por esto para calcular la heurística tengo en cuenta la fichas capturadas en este momento, las fichas capturadas del oponente, las cercos capturados por mi y por el contrario etc.

Una vez que sabemos los componentes que van a formar nuestra función de evaluación, nos queda hacer una gran cantidad de pruebas, dándole diferentes pesos para cada uno de estos componentes, hasta dar con la fórmula correcta. Es muy similar al ajedrez, cuando al peón le dan el valor 1, caballos y alfiles 3, torres 5, y reina 9, pero en nuestro caso le tenemos que dar pesos a las fichas capturadas, a los cercos controlados por nosotros, a la fichas en atari (en peligro de ser capturadas) etc. Para conseguir una buena función de evaluación, me surgieron algunas preguntas:

- **¿Intento hacerme fuerte en una esquina y me voy expandiendo solo desde esa esquina?**

Podría ser una opción, pero si mientras yo me voy haciendo fuerte en una esquina, el contrincante se va expandiendo por el resto del tablero, la partida estaría prácticamente perdida (si se juega bien), ya que aunque la zona de las esquina protegida por mí, sea prácticamente inaccesible para él, el domina el resto del tablero, y a la hora de expandirme a otros sitios para poder capturarlos, me resultara muy difícil, porque cuando ataque esa zona, el contrario la irá protegiendo a la vez.

- **¿En los primeros movimientos me expando por zonas estratégicas, y voy haciendo fuerte en estas zonas?**

Tras ver unos cuantos videos, de campeonatos del go, puedo decir que esta es la opción más adecuada, ir expandiéndote por el tablero al principio, si te ataca el contrario en una de estas zonas, ir defendiendo esa zona, y si no te ataca, vas haciéndote fuerte en ellas.

Para conseguir lo anterior, implementamos una jugada de apertura, en la cual, en los primeros movimientos, intentamos poner fichas en zonas estratégicas en el tablero que aseguren nuestro “futuro”, y a partir de esta jugada de apertura, vamos eligiendo la siguiente ficha con mini-max poda alfa-beta. La jugada de apertura, la he implementado de esta forma, después de ver los primeros movimientos de personas experimentadas en este juego, en los videos anteriormente dichos. Además de ayudarnos esta jugada de apertura, también nos vendrá bien para reducir el número de nodos evaluados para las primeras jugadas.

Antes de acabar, decir que conseguir que la función de evaluación “simule” lo que haríamos nosotros es muy complejo, y la toma de decisiones del ser humano es un proceso complejo, y se fundamenta en algo que los ordenadores no poseen. Para nosotros decidir si poner una ficha en un sitio o en otro, es algo más que darle puntos positivos y negativos, es decir nuestro

cerebro es tan complejo en la toma de decisiones, que es prácticamente imposible emularlo en un ordenador.

Para concluir, podemos decir que tratar el juego go como proyecto de fin de carrera ha sido bastante interesante. La aparente sencillez de sus reglas esconde un amplio y complejo abanico de combinaciones. Podemos aprender a jugar a dicho juego en 10 minutos, pero saber ser un buen estratega en el, te puede llevar toda una vida.

Capítulo 7. Trabajos futuros

Uno de los aspectos más interesante que se han quedados sin tratar, ha sido, dar opción a elegir el tamaño del tablero con el cual queremos jugar. Optamos trabajar con el tablero de 9*9 para simplificar un poco todas las posibles variaciones que se darían en un tablero de 19*19.

Una vez implementado el tablero de 19*19, deberíamos de aplicar nuevas mejoras al algoritmo mini-max, en el cual se reduciría aún más la evaluación de nodos, ya que hemos pasado de tener 81 posibles jugadas en el primer movimiento, a tener 361. En el tablero de 9*9 se puede jugar sin grandes esperas, con profundidad 4, pero con profundidad 5 la espera en los primeros movimientos es de horas. Si estamos en un tablero 19*19, si ponemos solo, profundidad 3, los nodos que evalúa son bastantes, y jugar con profundidad 4 con este tablero sería muy costoso de tiempo. Por todo lo anterior, si se implementara este tablero, tendríamos que hacer mejoras a mini-max para reducir bastante la evaluación de nodos, o buscar otros algoritmos que nos sean más eficientes. Además de que nuestra profundidad estaría muy limitada, por esto mismo, solo podríamos ver dos jugadas por delante, y un ser humano, en este juego puede prever muchas más cosas, por lo que sería mucho más fácil ganar al jugador máquina.

Capítulo 8. Bibliografía

- LEARN TO PLAY GO: A Master´s Guide To The Ultimate Game
    - Janice Kim
    - Jeong Soo-hyun

- LEARN TO PLAY GO: The Way Of The Moving Horse
    - Janice Kim
    - Jeong Soo-hyun

- FUNDAMENTOS DE INTELIGENCIA ARTIFICIAL
    - Miguel Ángel Cazorla Quevedo
    - Patricia Compañ Rosique
    - Francisco Escolano Ruiz
    - Ramón Rizo Aldeguer