**Parte B**

B.1.- Función de evaluación:

B1.1: El razonamiento que hemos seguido a la hora del desarrollo de la función de evaluación ha consistido principalmente en el análisis de diversas partidas y posibles estrategias que se nos ocurrían jugando contra la máquina.

En primer lugar desarrollamos un jugador muy básico, pero bastante efectivo dentro de lo que cabe, que se limita a puntuar el doble las fichas que están en el kalaha, lo que indica que es positivo conseguirlas. Este jugador realiza una resta de la puntuación de un jugador con la del contrincante.

Posteriormente hemos ido creando diversos jugadores basados en este jugador inicial, añadiendo ligeras modificaciones, muchas de ellas erróneas. Uno de ellos utiliza una función para comprobar hoyos vacíos y asignar puntuaciones según las fichas del contrincante, pero no lo hemos usado al final.

Por último nos hemos quedado con un jugador más básico que utiliza como estrategia el asignar puntuaciones más negativas a los hoyos cercanos al kalaha, lo que indica que no es una buena jugada, y que ha demostrado buena funcionalidad.

B1.2: Hace 2 cosas mal:

La primera es que intenta no terminar la partida porque esta puesto que se valore negativamente que la partida termine. Por tanto si tenemos una situación en la que tenemos que elegir 2 movimientos uno que implicaría que el movimiento del siguiente jugador termine la partida y otro que le entregue muchas fichas al otro jugador, se valorara más positivamente que entregue fichas al otro jugador, lo que puede llegar a ser muy negativo ya que puede hacer que se pierda la partida.

La segunda es que a la hora explorar nodos tiene en cuenta cual puede ser su mejor movimiento pero no tiene en cuenta si después eso puede beneficiar al adversario, por tanto si explora una profundidad más puede hacer movimientos que él considera buenos pero en verdad son malos ya que benefician más al adversario.

Una forma de arreglar esta situación es:

En el caso 1, que evalué positivamente si ha ganado y negativamente si ha perdido así, si ve que puede ganar vaya lo más rápidamente posible a esa situación. Y si ve que puede perder evite esa situación.

En el caso 2, no se nos ocurre más solución que directamente quitar esa situación.

B.2.- Minimax y Minimax a-b:

B2.1:

1. Básicamente tenemos una función que llama a minimax-1 que es donde se encuentra el minimax, que comprueba primero que no estemos explorando una profundidad mayor que la que debemos y que al generar los sucesores estos no sean nulos ya que si fuese así es que hemos llegado al final de la partida. Después hace un bucle en el que busca de los sucesores que hay cual es el mejor, para ello usa una llamada recursiva a sí mismo. Y una vez revisados todos los sucesores retorna el mejor de los que haya encontrado.

B2.2:

* Código entregado.
* El código viene a ser el mismo que tenemos para minimax, exceptuando ciertas cosas
  + Lo primero es que llamaos a una función minimax-a-b-1 que le tenemos que pasar alfa y beta, estos valores tendrán que ser máximo y mínimo.
  + Por otro lado a la hora de llamar de forma recursiva a minimax-a-b-1 cambiamos la alfa y la beta para que lleguen cambiadas y así solo tengamos que comprobar las alfas

B2.3:

B2.4: Si al algoritmo minimax con poda se le pasa el valor de los nodos explorados de forma ordenada, será más fácil para él descartar información ya que una vez encuentra que puede podar una rama la información le llega antes por el hecho de estar ordenada y tener toda la información en esa rama con valores peores.