3.1

| Profundidad | ID Jugador | Sin Poda | Con Poda | CP/SP |
|---|---|---|---|---|
| 1 | Jugador 1 | 0,02 | 0,01 | 81% |
| | Jugador 2 | 0,01 | 0,01 | |
| 2 | Jugador 1 | 0,21 | 0,20 | 92% |
| | Jugador 2 | 0,20 | 0,19 | |
| 3 | Jugador 1 | 3,93 | 1,26 | 28% |
| | Jugador 2 | 3,45 | 0,79 | |
| 4 | Jugador 1 | 72,68 | 7,90 | 10% |
| | Jugador 2 | 69,78 | 6,78 | |

Al utilizar la poda Alfa-Beta, los tiempos de respuesta del algoritmo disminuyen notoriamente, sobre todo cuando la profundidad aumenta. Esto se debe a que el algoritmo deja de analizar ramas que ya sabe que no le van a servir para así ahorrar tiempo de cómputo.

La razón teórica de esta diferencia de desempeño es que, en la medida que la profundidad del árbol va aumentando, la cantidad de estados crece exponencialmente. La poda Alpha-Beta hace que no se sigan evaluando ramificaciones que ya sabemos que no influirán en la decisión del jugador correspondiente. Esto dado que el oponente (Min) ya encontró valores menores para jugar en base a los estados que Max había dejado debajo suyo, o análogamente, el jugador Max ya encontró valores mayores entre los que dispuso Min en iteraciones anteriores, por lo que no tiene sentido seguir analizando las posibilidades de abajo, dado que no van a maximizar su valor en más de lo alcanzado hasta el minuto. Eliminar ramas de gran profundidad, sobre todo en estos problemas en que la cantidad de nodos hijos por cada padre es muy grande, permite ahorrar una cantidad de cómputo notable.

3.2

| Configuración | ID Jugador | Profundidad | Tiempo prom. | Ganador |
|---|---|---|---|---|
| 1 | Jugador 1 | 1 | 0,017 | 5 |
| | Jugador 2 | 1 | 0,018 | 5 |
| 2 | Jugador 1 | 1 | 0,016 | 1 |
| | Jugador 2 | 2 | 0,243 | 9 |
| 3 | Jugador 1 | 2 | 0,253 | 9 |
| | Jugador 2 | 1 | 0,014 | 1 |
| 4 | Jugador 1 | 1 | 0,014 | 0 |
| | Jugador 2 | 3 | 1,131 | 10 |
| 5 | Jugador 1 | 3 | 1,051 | 10 |
| | Jugador 2 | 1 | 0,014 | 0 |

A medida que aumenta la profundidad para un jugador Minimax en particular, tarda más en tomar una decisión, dado que recorre exponencialmente más nodos. Esto también significa que juega más estratégicamente ya que es capaz de mirar más a futuro los posibles resultados, en otras palabras, tiene una mayor capacidad de anticiparse a las jugadas de su contrincante. Al aumentar la profundidad comparativamente respecto al oponente, las posibilidades de ganar aumentan cada vez más dado que, como se mencionó anteriormente, el jugador con mayor profundidad tiene una mayor capacidad de evaluar a futuro y así anticiparse a las jugadas de su contrincante. Podemos ver que con una diferencia de 1 en profundidad el marcador suele terminar [9 – 1], mientras que ya con una diferencia de 2 en profundidad el marcador ya es [10 – 0].

3.3

| Configuración | ID Jugador | Profundidad | f(x) | Ganador | tiempo prom |
|---|---|---|---|---|---|
| 1 | Jugador 1 | 1 | amount | J2 | 0.016 |
| | Jugador 2 | 1 | values | | 0.018 |
| 2 | Jugador 1 | 1 | values | J1 | 0.023 |
| | Jugador 2 | 1 | amount | | 0.015 |
| 3 | Jugador 1 | 1 | amount | J2 | 0.013 |
| | Jugador 2 | 3 | values | | 0.991 |
| 4 | Jugador 1 | 3 | amount | J1 | 1,206 |
| | Jugador 2 | 1 | values | | 0,014 |

Según los resultados, la función de evaluación en base a los valores de cada celda es superior a la función de evaluación a base de la cantidad de círculos de cada jugador y además es más rápida para decidir. A una misma profundidad ganó siempre *score_circle_values* independiente de quién partió, mientras que a profundidades distintas ganó el de mayor profundidad.

Las diferencias en los tiempos de ejecución es difícil saber si se debe a las operaciones mismas de la evaluación o si se debe a que con la evaluación de valores se podan más ramificaciones, dado que las diferencias entre las situaciones de los jugadores es más notoria.

La evaluación basada en valores puede ser superior dado que tener más círculos en cada celda puede significar estar mejor posicionado para atacar en cada ronda, lo cual es ignorado por *score_circle_amount*.

3.4

Creé la función de evaluación *score_circle_opponent_min*, que lo que hace es contar la cantidad de círculos (valores) que tiene el jugador oponente y retornarlo en negativo (En realidad se le resta a 100 para que sea un score positivo). Creo que es una buena forma de evaluar este problema dado que el objetivo final del juego no es tener más fichas que el otro, sino dejar al otro sin fichas (independiente de la cantidad propia). Al final esta forma de evaluar apunta más directamente al objetivo para ganar el juego.

| Partida | ID Jugador | tiempo prom | f(x) | Ganador |
|---|---|---|---|---|
| 1 | Jugador 1 | 0,223 | opponent_min | J1 |
| | Jugador 2 | 0,221 | amount | |
| 2 | Jugador 1 | 0,210 | opponent_min | J1 |
| | Jugador 2 | 0,212 | amount | |
| 3 | Jugador 1 | 0,212 | opponent_min | J2 |
| | Jugador 2 | 0,211 | amount | |
| 4 | Jugador 1 | 0,212 | opponent_min | J1 |
| | Jugador 2 | 0,225 | amount | |
| 5 | Jugador 1 | 0,205 | opponent_min | J2 |
| | Jugador 2 | 0,209 | amount | |
| 6 | Jugador 1 | 0,206 | opponent_min | J1 |
| | Jugador 2 | 0,217 | amount | |
| 7 | Jugador 1 | 0,215 | opponent_min | J1 |
| | Jugador 2 | 0,217 | amount | |
| 8 | Jugador 1 | 0,224 | opponent_min | J1 |
| | Jugador 2 | 0,214 | amount | |
| 9 | Jugador 1 | 0,233 | opponent_min | J1 |
| | Jugador 2 | 0,239 | amount | |
| 10 | Jugador 1 | 0,222 | opponent_min | J2 |
| | Jugador 2 | 0,208 | amount | |

| Jugador | Tiempo Prom | Total ganados |
|---|---|---|
| 1 | 0,216 | 7 |
| 2 | 0,217 | 3 |

Los resultados que se muestran en la tabla corresponden a 10 partidas contra la función *score_circle_amount.* En esta ocasión ambos jugadores jugaron con profundidad 2 y el jugador con *score_circle_opponent_min* fue el que comenzó en todas las partidas. En esta tanda de 10 partidas, el jugador 1 (*score_circle_opponent_min*) ganó un 70% de las veces. Sin embargo, después de hacer esta tabla, se corrió dos veces más el código, ahora invirtiendo el orden de partida. Primero ganó el jugador con la función *score_circle_opponent_min* (jugador 2) con un 70%, y luego ganó el mismo con un 80% de las partidas, lo que comprueba la superioridad de la función *score_circle_opponent_min*. También se probó con otras profundidades obteniendo resultados parecidos.