NOMBRE: Bernardita Rosas

SECCIÓN:

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2613 — Inteligencia Artificial — 2' 2023

Tarea 3 – Respuesta Pregunta 3

3.1 Nota: Para mejorar los tiempos de ejecución, se usó la cota de alpha y beta según Algorithms Explained – minimax and alpha-beta pruning.
Profundidad = 1

Iteración	Ganador	Tiempo de ejecución (s)
Iteración 1	X	1.254412
Iteración 2	Empate	1.352571
Iteración 3	O	1.4495870000000002
Iteración 4	X	1.351591
Iteración 5	X	1.5625399999999998
Tiempo total	-	6.970701
Tiempo promedio	-	1.3941402

Profundidad = 3

Iteración	Ganador	Tiempo de ejecución (s)
Iteración 1	Empate	4.900741999999999
Iteración 2	Empate	3.6749989999999997
Iteración 3	Empate	7.844846
Iteración 4	Empate	5.962002999999999
Iteración 5	O	3.495728
Tiempo total	-	25.878317999999997
Tiempo promedio	-	5.175663599999999

Profundidad = 5

Iteración	Ganador	Tiempo de ejecución (s)
Iteración 1	Empate	239.366861
Iteración 2	Empate	79.27421100000001
Iteración 3	Empate	45.086659999999995
Iteración 4	Empate	40.990567
Iteración 5	Empate	50.23008
Tiempo total	-	454.948379
Tiempo promedio	-	90.9896758

Creo que el tiempo de ejecución aumenta debido a que al aumentar la profundidad, se aumenta también qué tan "al futuro" estamos anticipando para nuestra jugada. Claramente es diferente pensar que es lo mejor para la próxima jugada (a niveles generales) a qué es lo mejor considerando el resultado de las próximas 5 jugadas. Entonces, poniéndonos en la posición del supergato, en el que hay a lo más 9 jugadas disponibles, en profundidad 1 solo se tendrían tales 9 opciones, pero al sumergirnos a profundidad 5, cada una de esas 9 opciones cuenta con hasta 9 opciones más y así sucesivamente hay hasta $9^5 = 59049$ resultados posibles por jugada lo que claramente se va a demorarse más en procesar. Cabe destacar que me hace sentido que la cantidad de empates aumente al aumentar la profundidad ya que las IA toman decisiones más informadas y dependen menos de su suerte. En base a los datos, es bastante evidente el que a mayor profundidad mayor es el tiempo de ejecución, en particular, se puede observar que a una profundidad 5, este demora empíricamente, en promedio, 65 veces más en ejecutarse.

NOMBRE: Bernardita Rosas

SECCIÓN:

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2613 — Inteligencia Artificial — 2' 2023

Tarea 3 – Respuesta Pregunta

3.2

3.2.1

Profundidad = 3
Jugador 1:"X", funcion_puntaje=contar_tableros_diferencia
Jugador 2: "O", funcion_puntaje=contar_fichas_seguidas

Iteración	Ganador	Tiempo de ejecución (s)
Iteración 1	X	13.194603
Iteración 2	O	6.76565
Iteración 3	Empate	9.275166
Iteración 4	Empate	7.613937
Iteración 5	O	8.487704
Iteración 6	X	4.371115
Iteración 7	O	7.636415
Iteración 8	Empate	6.105988
Iteración 9	Empate	6.7496529999999995
Iteración 10	Empate	8.316576

Ganó contar_fichas_seguidas 3 veces y 2 veces contar_tableros_diferencia.

Profundidad = 3
Jugador 1:"X", funcion_puntaje=contar_fichas_seguidas
Jugador 2: "O", funcion_puntaje=contar_tableros_diferencia

Iteración	Ganador	Tiempo de ejecución (s)
Iteración 1	Empate	12.123942000000001
Iteración 2	Empate	8.258804
Iteración 3	X	5.0479210000000005
Iteración 4	X	5.754296
Iteración 5	Empate	5.527893000000001
Iteración 6	X	10.249648
Iteración 7	Empate	8.721661000000001
Iteración 8	X	4.624715
Iteración 9	Empate	4.867808
Iteración 10	Empate	7.900451

Ganó contar_fichas_seguidas 4 veces y 0 veces contar_tableros_diferencia.

Una función de puntaje en minimax se encarga de estimar que tan bueno es el resultado propio, o malo el del oponente para priorizar tales jugadas que maximicen las probabilidades de ganar, funcionando de manera similar a una heurística pero actuando a cierta profundidad ya que de lo contrario sería demasiado demoroso. En cuanto a los resultados obtenidos: 7 veces ganó contar_fichas_seguidas, 2 vez contar_tableros_diferencia, mientras que se empataron los 11 juegos restantes, demostrando una ventaja de la función contar_fichas_seguidas y de los jugadores que comienzan primero.
Me parece bastante razonable el que exista una ventaja moderada (siguen existiendo muchos empates) en la nueva función propuesta ya que hay varios factores que no considera, por ejemplo la visión más general del tablero (solo lo revisa al ganar pero no prioriza subtableros sobre otros en base a su posición) además de que cuenta parejas dentro de los subtableros sin considerar si estos se encuentran bloqueados por el oponente (bajo esta función, xxo, aporta en puntaje) y tampoco considera pares no seguidos de fichas que si pueden (x_x). Todos estos factores planeo tomarlos en cuenta al construir mi propia función que solo va a considerar los subtableros que no se encuentren obstruidos por el adversario. Aún así logra superar a la función inicial contar_tableros_diferencia, lo que sí me hace sentido ya que esta realmente no me parece que aporte demasiado a medir que tan bien esta la posición del jugador actual.
Por otro lado, me hace bastante sentido que el jugador que comienza gane con mayor frecuencia (por ejemplo la función contar_tableros_diferencia gana dos veces cuando es primero pero ninguno al ser segundo, contar_fichas_seguidas también aumenta su frecuencia de ganar) ya que pueden ejercer una jugada más muchas veces, lo que les da una ventaja que siempre es valiosa.

3.2.2

Como ya anticipé, mi función toma varias cosas en cuenta que no se tomaron en cuenta en la función contar_fichas_seguidas. Lo principal respecto a mi función es que solo va a sumar puntos en subtableros no bloqueados por el oponente tanto a nivel macro como a nivel micro (dentro de cada tablero). Por cada posibilidad (es decir si el mismo subtablero tiene posibilidad de hacer una diagonal o por columna, se cuenta dos veces) que tiene un subtablero de ganar, se suma el puntaje de sus fichas relevantes (uno por cada pareja de fichas no bloqueadas). Esto creo que funcionará mejor al no considerar ningún intento frustrado por ganar. En particular, suma 5 por cada subtablero ganado relevante (es decir que no se encuentra bloqueado por el oponente, cabe mencionar que, si puede ganar por diagonal y por fila, entonces este puntaje se cuenta dos veces, una por cada forma en la que puede ganar). Además, por cada fila (columna o diagonal) que no se encuentra bloqueada por el oponente (pero no se ha ganado el subtablero), se suma un punto por cada pareja (un simbolo puede ser parte de más de una pareja) no bloqueada que podría potencialmente ganar el subtablero relevante (solo se cuenta si es que el subtablero no está ganado).

Profundidad = 3

Iteración	Ganador	Tiempo de ejecución (s)
Iteración 1	X	16.877012999999998
Iteración 2	X	15.389660999999998
Iteración 3	O	15.16982
Iteración 4	X	9.114062
Iteración 5	O	18.125541000000002
Iteración 6	O	7.165268
Iteración 7	O	7.745771
Iteración 8	O	16.810584
Iteración 9	Empate	11.401190999999999
Iteración 10	O	11.401190999999999

Cabe mencionar que contar_fichas_seguidas correspondió al primer jugador en las primeras 5 iteraciones (siempre fue la letra X), mientras que mi_funcion correspondió al primer jugador entre la iteración 6 y 10 (Letra O). Entonces, claramente mi_funcion resultó la ganadora con 6 resultados favorables, tres en contra y dos empates. Sobre todo al corresponder al jugador 1, aunque vale notar que el tiempo de ejecución aumentó considerablemente debido a los cálculos extra. Analizando aún más, me parece que otra forma para analizar la calidad de un tablero pudiese ser la cantidad de lugares en los que si se coloca una ficha entonces se gana el subtablero relevante (entre más mejor).

Al menos 10 juegos en que un jugador Minimax utilice la función de puntaje implementada en esta actividad usando una profundidad de 1 y el otro jugador Minimax debe usar la mejor función de valor obtenida en la actividad anterior con una profundidad de 5. Queda a tu criterio cual de las dos comienza.

Iteración	Ganador	Tiempo de ejecución (s)
Iteración 1	Empate	223.411556
Iteración 2	X	193.492712
Iteración 3	Empate	436.66864899999996
Iteración 4	Empate	393.076869
Iteración 5	X	142.666218
Iteración 6	O	246.39271100000002
Iteración 7	X	714.298755
Iteración 8	X	618.689661
Iteración 9	Empate	11.401190999999999
Iteración 10	X	755.781559

Notar que O corresponde a mi_función con profundidad 1 y X corresponde a contar_fichas_seguidas con profundidad 5. Siempre fue primer jugador contar_fichas_seguidas Considerando la gigante ventaja que supone tener una mayor profundidad ya que permite ver más adelante en la jugada, como se discutió en la pregunta 2.1, me parece que mi función actúo bastante bien, además que tenía la desventaja en cuanto a no empezar e incluso ganó una vez.

Iteración	Ganador	Tiempo de ejecución (s)
Iteración 1	O	159.201579
Iteración 2	O	339.36239600000005
Iteración 3	O	346.67218099999997
Iteración 4	O	364.60483600000003
Iteración 5	X	550.515476c
Iteración 6	O	358.476286
Iteración 7	Empate	376.332328
Iteración 8	O	697.107946
Iteración 9	O	1000.430059
Iteración 10	O	718.40625

En este caso, mi_funcion correspondió a letra O, con profundidad igual a 5, mientras que contar_fichas_seguidas correspondió a X con profundidad 1.
Me parece que mi_funcion es una mejora importante sobre contar_fichas_seguidas, evidenciándose en que bajo mismas condiciones de inicio y profundidad (ya que los fui cambiando según se señaló) mi_funcion ganó 15 veces, contar_fichas_seguidas 9 veces, mientras que se empataron las restantes 6 veces. Ahondé en porqué creo que mi función es mejor (revisar más arriba) pero lo esencial es que se asegura de contabilizar fichas como una mejor solución si y solo si estas no se encuentran frustradas por su oponente ya sea a nivel macro o micro.
Luego, es evidente que a mayor profundidad, mayor es la posibilidad de ganar ya que, como se mencionó en 3.1, se exploran escenarios posibles y su valor en jugadas posteriores. Claramente si uno planifica pensando en su posición en 5 jugadas más adelante va a tener una ventaja sobre otra persona que solo considera el estado (aproximado) de la próxima jugada, la planificación a largo plazo recompensa, aunque puede haber casos en lo que no hace tanto la diferencia. Esto se evidencia claramente ya que ambos algoritmos ganaron cuando se usaron con profundidad 5 y su oponente con profundidad 1, mi_funcion (8/10 v/s 1/10) y contar_fichas_seguidas(5/10 v/s 1/10), incluso cuando mi_funcion resulto ser una mejor medidora de puntajes a profundidad 3.