MEMORIA PRÁCTICA 1 - B

Juan Larrondo y Abril Palanco

Grupo: 1311

Parte B: Búsqueda con adversarios

B.1 Búsqueda de soluciones en un entorno estático

El objetivo de este apartado es aplicar los métodos de búsqueda implementados en la parte A a un tablero estático (sin adversario) y ver cómo se comportan cada uno de ellos con diferentes casos del juego othello.

B.1.1

En todos los casos, casi todos encuentran el mismo camino (el mejor, pues es el que encuentra BFS), con la excepción de DFS, que encuentra un camino malo, pero es el algoritmo más rápido en terminar de todos.

B.1.2

Se puede ver que los A* varían mucho en tiempo dependiendo de la calidad de la heurística. Una buena heurística hace que encuentren una solución muy buena (la mejor en este caso) en un tiempo muy parecido a DFS.

```
juan@LAPTOP-UF2DBJK9:~/A3Q1/IA/pacman$ python othello.py
Initial board after 15 iterations:
  abcdef
1 .W__..
2 BWW_BW
3 .BWWW_
4 W_WWB.
5 _WBB..
6 _.B.W.
Problem to be solved: <class '__main__.OthelloSearchProblemOneCorner'>
BFS found a path of 1 moves in 0:00:00.001558: [('a', 6)]
DFS found a path of 3 moves in 0:00:00.000725: [('f', 3), ('b', 6), ('a', 6)]
UCS found a path of 1 moves in 0:00:00.001474: [('a', 6)]
Amin found a path of 1 moves in 0:00:00.000407: [('a', 6)]
Amax found a path of 1 moves in 0:00:00.003332: [('a', 6)]
Asum found a path of 1 moves in 0:00:00.002661: [('a', 6)]
Acomplex found a path of 1 moves in 0:00:00.000498: [('a', 6)]
```

```
juan@LAPTOP-UF2DBJK9:~/A3Q1/IA/pacman$ python othello.py
Initial board after 15 iterations:
  abcdef
1 .WWWWW
2 ..WWWW
3 ._WW.B
4 _WWWB.
5 ...B..
6 ..BW_.
Problem to be solved: <class '__main__.OthelloSearchProblemTwoCorners'>
BFS found a path of 3 moves in 0:00:00.005023: [('a', 4), ('a', 5), ('a', 6)]
DFS found a path of 8 moves in 0:00:00.001786: [('e', 6), ('f', 4), ('b', 3), ('c', 5),
 ('a', 3), ('b', 6), ('b', 5), ('a', 6)]
UCS found a path of 3 moves in 0:00:00.005420: [('a', 4), ('a', 5), ('a', 6)]
Amin found a path of 3 moves in 0:00:00.005884: [('a', 4), ('a', 5), ('a', 6)]
Amax found a path of 3 moves in 0:00:00.004576: [('a', 4), ('a', 5), ('a', 6)]
Asum found a path of 3 moves in 0:00:00.004943: [('a', 4), ('a', 5), ('a', 6)]
Acomplex found a path of 3 moves in 0:00:00.001858: [('a', 4), ('a', 5), ('a', 6)]
```

```
juan@LAPTOP-UF2DBJK9:~/A3Q1/IA/pacman$ python othello.py
Initial board after 15 iterations:
  abcdef
1 ._W...
2 B_W.__
3 WWWBW_
4 _WBBB.
5 _WWBWB
6 __B._W
Problem to be solved: <class '__main__.OthelloSearchProblemAllCorners'>
BFS found a path of 5 moves in 0:00:02.554522: [('f', 2), ('a', 1), ('a', 6), ('e', 2), ('f', 1)]
DFS found a path of 16 moves in 0:00:00.002509: [('e', 2), ('f', 3), ('a', 6), ('d', 1), ('b', 2), ('a', 4),
 ('e', 6), ('b', 6), ('e', 1), ('d', 2), ('a', 5), ('f', 4), ('b', 1), ('f', 2), ('f', 1), ('a', 1)]
UCS found a path of 5 moves in 0:00:02.399530: [('f', 2), ('a', 1), ('a', 6), ('e', 2), ('f', 1)]
Amin found a path of 5 moves in 0:00:02.905478: [('f', 2), ('a', 1), ('a', 6), ('e', 2), ('f', 1)]
Amax found a path of 5 moves in 0:00:02.759558: [('f', 2), ('a', 1), ('a', 6), ('e', 2), ('f', 1)]
Asum found a path of 5 moves in 0:00:02.764005: [('f', 2), ('a', 1), ('a', 6), ('e', 2), ('f', 1)]
Acomplex found a path of 5 moves in 0:00:00.080637: [('a', 6), ('a', 1), ('f', 2), ('e', 2), ('f', 1)]
```

B.2 Funciones de evaluación heurística para búsqueda con adversarios

El objetivo es encontrar buenas heurísticas para el juego reversi. Para ello hemos estudiado qué métricas son las más significativas en el juego. Estas son:

- Movilidad o mobility - M: Los movimientos legales en el tablero actual. Muy relevante, sobre todo al principio.
- Puntuación por casilla:
    - Esquinas o corners - C: Posiciones muy importantes a lo largo de la partida.
    - X_squares - X: Son las piezas que rodean a las esquinas. Cogerlas demasiado pronto puede dar ventaja al contrincante para coger las esquinas.
    - Lados o edges - E: Son menos estables que las esquinas, pero más que otras casillas más expuestas.
- Estabilidad o stability - S: Una pieza estable es una que no te pueden quitar. Tiene importancia durante toda la partida, pero más en las fases finales.

  Para el cálculo de la estabilidad empleamos una forma que parece ser muy usada en heurísticas para el reversi. Consiste en que, para ver si una ficha es estable hay que ver que en todas direcciones (horizontal, vertical y diagonal) no haya casillas vacías o del contrincante antes de llegar al límite del tablero o a otra casilla estable.

- Puntuación total - P: Diferencia de puntuación entre el jugador y el contrincante. Sólo relevante en las últimas etapas del juego.

La relevancia de las métricas cambia a lo largo de la partida. Lo cuál es algo a tener en cuenta.

Al hacer las heurísticas, probamos varias combinaciones de uso para todas las métricas, con diferentes pesos (incluyendo todos a 1) y variando (o no) los pesos dependiendo de la fase de la partida.

Torneo 1

Las primeras 3 soluciones que mejor nos funcionaron (las entregadas para el primer torneo) usan M, C y S normalizadas, ajustando, o no (MCS), los pesos de forma estática (MCSW) o dinámica según la fase de la partida (MCSTW). Demostraron ser buenas estrategias, ya que ganaban a la heurística que devuelve valores aleatorios (rand) un 85-90% de las veces y nos permitió quedar 7 de 57 participantes en el torneo (percentil 87.72).

Consideramos importante normalizar las métricas, pues la diferencia entre esquinas puede ser un máximo de 4, cuando en movilidad puede ser mayor, mientras que la importancia de las esquinas en determinada fase de la partida puede ser mayor.

Hicimos 3 pruebas para ver el porcentaje de victoria sobre rand y entre ellos. Las pruebas variaban el número de veces que juegan entre ellos y la profundidad de las futuras jugadas a considerar. Los resultados son:

Prueba 1: 200 partidas entre cada uno (100 en blancas y 100 en negras) y profundidad de 1. En esta prueba los resultados son que la heur. que más gana a las otras es MCSTW. Mientras que el nº de victorias contra rand es similar, entre el 86-89%.

```
Results for tournament where each game is repeated 200=100x2 times, alternating colors for each player
         total:  1_MCS   2_MCSW  3_MCSTW 4_rand
1_MCS    278:    ---     100     0       178
2_MCSW   272:    100     ---     0       172
3_MCSTW  573:    200     200     ---     173
4_rand   76:     22      27      27      ---
```

Prueba 2: 150 partidas entre cada uno y profundidad de 2. Entre nuestras heur., se ganan el mismo nº de veces entre ellas, 75 cada una. A rand ganan todas entre el 96-98.7%. Este es un muy buen resultado, pero hay que tener en cuenta que es para una profundidad de 2.

```
Results for tournament where each game is repeated 150=75x2 times, alternating colors for each player
         total:  1_MCS   2_MCSW  3_MCSTW 4_rand
1_MCS    294:    ---     75      75      144
2_MCSW   298:    75      ---     75      148
3_MCSTW  296:    75      75      ---     146
4_rand   12:     6       2       4       ---
```

Prueba 3: 100 partidas entre cada uno y profundidad 3. De nuestras heur. la que más pierde es MCSTW, que pierde todas las partidas contra otras heurísticas. El porcentaje de victorias contra rand es ahora de entre 92-96%.

```
Results for tournament where each game is repeated 100=50x2 times, alternating colors for each player
         total:  1_MCS   2_MCSW  3_MCSTW 4_rand
1_MCS    245:    ---     50      100     95
2_MCSW   246:    50      ---     100     96
3_MCSTW  92:     0       0       ---     92
4_rand   17:     5       4       8       ---
```

Prueba 4: 60 partidas entre cada uno y profundidad de 4. Nuestras heur. se ganan el mismo nº de veces entre ellas. Mientras que a rand le ganan entre 71.6-86.7%, menos que en anteriores pruebas y siendo, claramente, MCSTW la que peor juega.

```
Results for tournament where each game is repeated 60=30x2 times, alternating colors for each player
         total:  1_MCS   2_MCSW  3_MCSTW 4_rand
1_MCS    112:    ---     30      30      52
2_MCSW   111:    30      ---     30      51
3_MCSTW  103:    30      30      ---     43
4_rand   34:     8       9       17      ---
```

Concluimos que dependiendo de la profundidad de las jugadas futuras a tener en cuenta, los resultados cambian bastante, por lo que entregamos las 3 heurísticas, sin sustituir ninguna por pequeñas variaciones de otras.

Torneo 2

Para el segundo torneo cambiamos un poco los pesos, la forma de normalizar y el ajuste de pesos en las fases de la partida. No observamos una clara mejora con respecto a las anteriores heurísticas. Por un error en la entrega, no pudimos participar en el torneo.

Hicimos las mismas pruebas que en el primer torneo (a excepción de la 4º, con profundidad 4, ya que parece ser que los torneos no usan tanta profundidad), obteniendo resultados prácticamente iguales, entre el ~85% y el ~98% de victorias contra rand y misma relación de victorias entre nuestras heurísticas.

```
Results for tournament where each game is repeated 200=100x2 times, alternating colors for each player
        total:  opt1_MCS        opt2_MCSW       opt3_MCSTW      opt4_rand
opt1_MCS        271:    ---     100     0       171
opt2_MCSW       278:    100     ---     0       178
opt3_MCSTW      572:    200     200     ---     172
opt4_rand       79:     29      22      28      ---
```

Prueba 1 - Victoria vs rand: 85.5-89%

```
Results for tournament where each game is repeated 150=75x2 times, alternating colors for each player
        total:  opt1_MCS        opt2_MCSW       opt3_MCSTW      opt4_rand
opt1_MCS        294:    ---     75      75      144
opt2_MCSW       294:    75      ---     75      144
opt3_MCSTW      292:    75      75      ---     142
opt4_rand       20:     6       6       8       ---
```

Prueba 2 - Victoria vs rand: 94.7-96%

```
Results for tournament where each game is repeated 100=50x2 times, alternating colors for each player
        total:  1_MCS   2_MCSW  3_MCSTW 4_rand
1_MCS   245:    ---     50      100     95
2_MCSW  248:    50      ---     100     98
3_MCSTW 92:     0       0       ---     92
4_rand  15:     5       2       8       ---
```

Prueba 3 - Victoria vs rand: 92-98%

Entrega final

En la entrega final hemos modificado los pesos ligeramente, cambio que ha afectado directamente a los porcentajes de victoria.

```
Results for tournament where each game is repeated 200=100x2 times, alternating colors for each player
        total:  1_JA1   2_JuanAbril2    3_AbrilJuan3    4_rand
1_JA1   269:    ---     100     0       169
2_JuanAbril2    287:    100     ---     0       187
3_AbrilJuan3    574:    200     200     ---     174
4_rand  70:     31      13      26      ---
```

Prueba 1 - Victoria vs rand: 84,5-93,5%

```
Results for tournament where each game is repeated 150=75x2 times, alternating colors for each player
        total:  1_JA1   2_JuanAbril2    3_AbrilJuan3    4_rand
1_JA1   296:    ---     75      75      146
2_JuanAbril2    297:    75      ---     75      147
3_AbrilJuan3    299:    75      75      ---     149
4_rand  8:      4       3       1       ---
```

Prueba 2 - Victoria vs rand: 97,3-99,3%

```
Results for tournament where each game is repeated 100=50x2 times, alternating colors for each player
        total:  1_JA1   2_JuanAbril2    3_AbrilJuan3    4_rand
1_JA1   248:    ---     50      100     98
2_JuanAbril2    247:    50      ---     100     97
3_AbrilJuan3    83:     0       0       ---     83
4_rand  22:     2       3       17      ---
```

Prueba 3 - Victoria vs rand: 83-98%

B.3 Minimax con poda

El problema a resolver es implementar la clase “MinimaxAlphaBetaStrategy”, con MinimaxStrategy como posible punto de partida.

Para esto hay que ir pasando entre nodos la tupla de valor alpha y beta e ir cambiando su valor según estemos en un nodo min o uno max. En este caso, allá donde se use la variable minimax_value en MinimaxStrategy, se sustituye por el correcto uso de la tupla alpha, beta. Además, antes de terminar cada evaluación de un sucesor, si alpha es más grande que beta, no consideramos más sucesores (Poda alpha-beta), de forma que se reduce en gran medida el tiempo de ejecución total.

Nuestra implementación de MinimaxAlphaBetaStrategy sigue el siguiente pseudocódigo:

```
class MinimaxAlphaBetaStrategy:

 function next_move(state):
        alpha, beta, move ← _max_value(state, profundidad_max, -∞, +∞)
        return move

 function _min_value(state, depth, alpha, beta):
        if (estado_terminal) or (depth == 0):
                beta ← heuristic.evaluate(state)
                mejor_movimiento ← None
        else:
                value ← +∞
                for each successor in generate_successors(state):
                        successor_alpha, _, _ ← _max_value(successor, depth-1, alpha, beta)
                        if (successor_alpha < value):
                                value ← successor_alpha
                                if (value < beta): beta ← value
                                mejor_movimiento ← successor
                        if (alpha >= beta): break // Poda alfa-beta
        return (alpha, beta, mejor_movimiento)

 function _max_value(state, depth, alpha, beta):
        if (estado_terminal) or (depth == 0):
                alpha ← heuristic.evaluate(state)
                mejor_movimiento ← None
        else:
                value ← -∞
                for each successor in generate_successors(state):
                        _, successor_beta, _ ← _min_value(successor, depth-1, alpha, beta)
                        if (successor_beta > value):
                                value ← successor_beta
                                if (value > alpha): alpha ← value
                                mejor_movimiento ← successor
                        if (alpha >= beta): break // Poda alfa-beta
        return (alpha, beta, mejor_movimiento)
```

Bibliografía Reversi

https://kartikkukreja.wordpress.com/2013/03/30/heuristic-function-for-reversiothello/

https://www.researchgate.net/publication/353551698_An_Extensible_and_Modular_Design_and_Implementation_of_Monte_Carlo_Tree_Search_for_the_JVM

https://www.ffothello.org/livres/beginner-Randy-Fang.pdf

https://www.coolmathgames.com/es/blog/c%C3%B3mo-jugar-estrategia-y-conceptos-b%C3%A1sicos-de-reversi

https://ceur-ws.org/Vol-1107/paper2.pdf