



UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones

Memoria Práctica 3

Búsqueda con Adversario (Juegos) - El Parchis

Inteligencia Artificial

Torres Ramos, Juan Luis

2°C - Grupo 2

5 junio de 2022

1. Análisis del Problema

Tenemos el juego del parchís donde diseñaremos técnicas de búsqueda con adversario en dicho entorno. Las modificaciones a tener en cuenta son: cada jugador juega con 2 colores alternos (0: amarillo y rojo | 1: azul y verde) y además que los dados se van eligiendo entre un conjunto de dados disponibles, es decir, el dado no es azar.

Para desarrollar las técnicas de búsqueda con adversarios nos basaremos en la implementación de las estrategias minimax y la poda alpha-beta del árbol del juego. Primero desarrollamos la MinMax ya que la de poda es una variación de esta última. Para la entrega estaremos utilizando la poda.

2. Descripción de la Solución Planteada.

Primero tenemos nuestra función think() donde definiremos cómo interactúa nuestro agente. Para ello con un identificador id manejaremos varias heurísticas, para poder compararlas entre ellas y luego compararlas con los ninjas.

La id = 0 será la primordial a la hora de evaluar la práctica, será nuestra heurística final.

Luego he implementado primero la función MinMax y la función Poda-Alpha-Beta. Los pasos para su implementación son los siguientes:

Funcion MinMax:

El procedimiento que seguido para implementar la función es la siguiente:

1. Primero compruebo la condición `profundidad == profundidad max` || `actual.gameOver()`, en caso de llegar al final o del que juego acabe devolvemos la heurística
2. inicializamos los valores de `last_c_piece`, `last_id_piece`, `last_dice`, `valor`, `aux` respectivamente
3. Genero mi hijo llamando a la función `GenerateNextMove`, al igual que en tutorial
4. Mientras `hijo != actual`
 - a. genero `valor aux` a través llamada recursiva de `MinMax` aumentando la profundidad +1
 - b. Si `actual.getCurrentPlayerid() == jugador`, es decir, es un nodo max
 - i. si `aux > valor`, `valor` siendo inicializado a `menosinf`, actualizamos `valor` y cuando `profundidad == 0` actualizo la acción
 - c. Para el caso que `actual.getCurrentPlayerid() != jugador` es un nodo min
 - i. si `aux < valor`, `valor` siendo inicializado a `massinf`, actualizamos `valor` y cuando `profundidad == 0` actualizo la acción
 - d. Por ultimo genero otro hijo
5. Devolvemos `valor`

Función Poda-Alpha-Beta:

Dicha función es similar a la MinMax, sin embargo utilizamos `alpha` y `beta` en vez de `valor`: Cambian los siguientes apartados

....

- a. genero `valor aux` a través llamada recursiva de `PodaAlpahBeta` aumentandole la profundidad +1.
- b. Si `actual.getCurrentPlayerid() == jugador`, es decir, es un nodo max
 - i. si `aux > alpha`, `alpha` siendo inicializado a `menosinf`, actualizamos `alpha` y cuando `profundidad == 0` actualizo la acción
 1. Si `alpha >= beta` podo el arbol, hago un `break`.
- c. Si `actual.getCurrentPlayerid() != jugador`, es decir, es un nodo min
 - i. si `aux < beta`, `beta` siendo inicializado a `masinf`, actualizamos `beta` y cuando `profundidad == 0` actualizo la acción
 1. Si `alpha >= beta` podo el arbol, hago un `break`.
6. En caso de que estemos en un nodo max devolvemos `alpha`, si no devolvemos `beta`.

Luego, he probado dichas funciones con la heurística de prueba llamada Valoración Test, al enfrentarla ambas contra el ninja 1 ganaban sin problema, pero a la hora de enfrentarlas a el ninja 2 perdieron estrepitosamente. Ahora me voy a centrar en implementar mi propia heurística a la que voy a llamar Valoración, los pasos para implementarla son los siguientes

Valoración:

Para la heurística me he centrado en valorar positivamente aquellos estados donde a mi jugador le convenga estar mi jugador recorriendo mi color y fichas respectivas. También por el otro lado he tenido en cuenta mi oponente, valorando negativamente los estados que a él que le convenga:

Jugador:

- valoro positivamente si la casilla es goal. (2000)
- valoro negativamente si la casilla es home (50)
- valoro positivamente si se come una ficha con isEatingMove() (150)
- valoro positivamente si hace muros de su respectivo color (200)
- Probé a valorar aquel jugador más adelantado, pero no veía mejoras.(50)

Oponente

- valoro negativamente si la casilla es goal. (-2000)
- Probé a valorar negativamente si come, pero a veces rinde peor.
- valoro negativamente si va a una casa (-250)
- valoro negativamente si hace muro (-100)

Haciendo varias pruebas de la práctica ejecutando

```
./bin/Parchis --p1 NINJA 1 "NINJA" --p2 AI 0 "J1"
```

por lo que juego las fichas verdes y azules

Haciendo pruebas, veo un comportamiento más o menos inteligente por parte del jugador, hace muros, tiende a ir a casillas casa y cuando tiene la oportunidad come las fichas a su alcance. Eso sí, hay veces que hace movimientos poco convenientes, o no come las fichas que debería comer.

Vence al NInja 1, he hecho varias pruebas y por lo general lo vence, ha habido veces que también ha perdido con el ninja 1, pero las veces que le gana le mete una paliza el jugador

Dejo aquí una captura de una de las victorias de mi jugador al ninja 1:

```

285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302

```

if (estado.isEat
 puntuacion_j

 if(estado.isWall
 puntuacion_juga

 if (estado.isSaf
 puntuacion_j

 if(estado.distan
 min_distancia

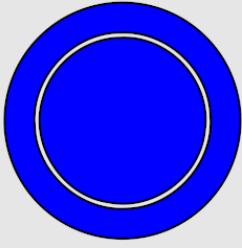
 id_ficha_mas_a

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TE

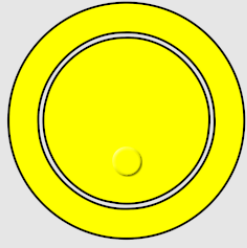
Valor MiniMax: 1e+10 Accion: Azul 1 6
 Movimiento elegido: Azul 1 6
 301 MOVED sent
 Move: 215 Azul 1 6
 New move received. Updating GUI
 Last action: Azul 1 6
 last dice: -1
 last dice: -1

 Turno: 215
 Jugador actual: 1 (J1)
 Color actual: Azul

 Realizo un movimiento automatico
 Valor MiniMax: 1e+10 Accion: Azul 1 3
 Movimiento elegido: Azul 1 3
 301 MOVED sent
 Move: 216 Azul 1 3
 New move received. Updating GUI
 Last action: Azul 1 3
 last dice: -1
 ++++++
 La partida ha terminado
 Ha ganado el jugador 1 (Azul)
 ++++++
 □



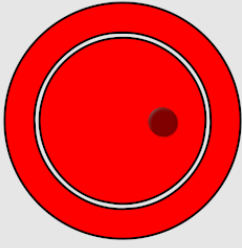
16	17	18
15	19	
14	20	
13	21	
12	22	
11	23	
10	24	
9	25	



33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18
34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49

8	7	6	5	4	3	2	1
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32

68	67	66	65	64	63	62	61
69	70	71	72	73	74	75	76
77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92



50	49	48	47	46	45	44	43
51	52	53	54	55	56	57	58
59	60	61	62	63	64	65	66
67	68	69	70	71	72	73	74

