PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

KNIGHT CHESS

ERICK JHONATAN MAULEN TORO
LUCAS TOMAS AGULLO VIVEROS
MATIAS ALONSO ESPINOZA ARANGUIZ
BENJAMIN JESUS ROJAS DELPINO

ESTRUCTURAS DE DATOS AVANZADAS Y ALGORITMOS

Septiembre, 2020

Índice

Lista de Figuras

1. Modelado

Para el modelado de este problema se trabajó mediante una entrada en formato JSON, el cual se carga en en el siguiente código.

```
jsn = json.loads(sys.argv[1])
```

Figura 1: Cargando los datos.

Para luego pasarlo a OwervanzSearchTree el cual llamará el método MCTS para obtener la id del caballo a mover y donde se moverá.

```
ovc = OwervanzSearchTree(jsn)
result = ovc.mcts()
key = list(result)[0]
move = str(result[key][0]) + "," + str(result[key][1])

action = {
    "knight_id": int(key),
    "knight_movement": moveDict[move]
}
```

Figura 2: Obtener resultados.

Se decidió recrear un estado mediante un array de Numpy, para poder tener una simulación más concreta y certera. Según como se ejecute, player 1 o player 2, nuestras piezas podrán variar entre 100 y 115 o 200 y 2015 . Siguiendo en las restricciones del problema, las transiciones recibirán la pieza a mover y el destino donde deberá ir. Por último la representación del nodo se ilustra en la siguiente imagen:

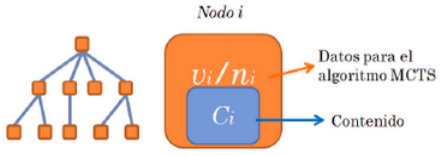

Figura 3: Representacion de Nodo

En donde se podrá encontrar el estado correspondiente según las acciones que ha realizado, su número de visitas, su valor del coeficiente UCT, la recompensa, entre varios.

2. Estrategia

La estrategia que se ocupó en este desafío fue aplicar Monte-Carlo Tree Search (MCTS) que es el primer método de búsqueda que no requiere una función de evaluación de posición,consiste en cuatro pasos principales, repetidos tantas veces como tiempo disponible haya. En una de las iteraciones se parte de la situación inicial de la partida (situación de la partida en el momento de la simulación), pero se conserva el árbol MCTS, completando durante las distintas fases y simulaciones.

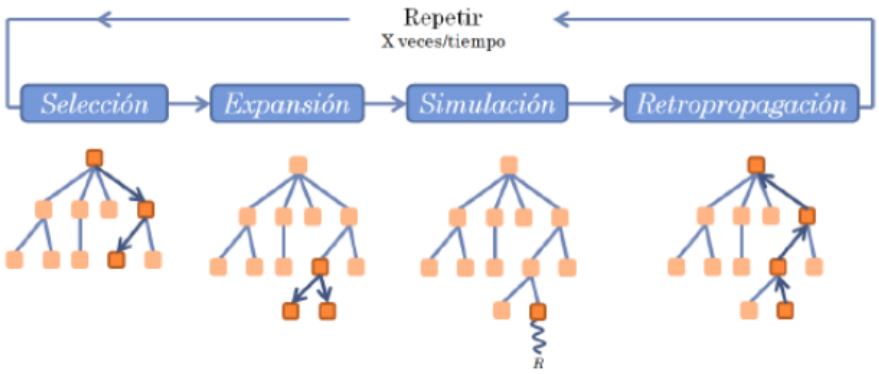

Figura 4: Etapas MCTS.

A continuación se mostrará el pseudocódigo correspondiente al MCTS:

Algoritmo 4.1 Pseudocódigo del MCTS

```
Datos: nodoRaiz
Resultado: mejorMovimiento
MientrasQue (haya_tiempo) hacer
    nodoActual ← nodoRaiz
    /* Selección */
    MientrasQue (nodoActual ∈ A) hacer
        nodoActual ← Seleccionar(nodoActual)
    Fin
    /* Expansión del nodo */
    nodoActual ← Expandir(nodoActual)
    /* Simulación de una partida*/
    R ← JugarPartidaSimulada(nodoActual)
    /* Retropropagación del resultado */
    MientraQue(nodoActual ∈ A) hacer
        Retropropagación(nodoActual, R)
        nodoActual = nodoActual.padre
    Fin
Fin
Devuelve mejorMovimiento = MejorHijo(nodoRaiz)
```

Figura 5: Pseudocodigo MCTS.

Como política usamos el Upper Bound Confidence for Trees (UCT), que consiste en una fórmula matemática que estima el valor máximo que puede obtener un nodo "optimista" al ser simulado.

$$UCT = \overline{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}}$$

Figura 6: Politica de evaluacion UCT.

Donde:

$\overline{X_j}$ Corresponde al promedio de simulaciones del nodo j.
n Simulaciones del padre de j.
n_j Simulaciones de j.
C_j Nivel de optimismo, el cual se ajusta para controlar la exploración o explotación.

Figura 7: Datos de UCT .

El primer término prioriza buenos resultados en explotación y el segundo termino prioriza buenos resultados en exploración.

Por otro lado ocupamos una estrategia para mejorar la recompensa, la cual fue sacado de la wikipedia de programación de ajedrez. Dónde cada posición corresponderá a un puntaje, para el jugador actual será positivo, y para el enemigo será negativo. Además de esta evaluación, tenemos una que será la diferencia de las piezas, dónde será positiva si favorece al jugador, caso contrario negativa.

```
-50,-40,-30,-30,-30,-30,-40,-50,
-40,-20,  0,  0,  0,  0,-20,-40,
-30,  0, 10, 15, 15, 10,  0,-30,
-30,  5, 15, 20, 20, 15,  5,-30,
-30,  0, 15, 20, 20, 15,  0,-30,
-30,  5, 10, 15, 15, 10,  5,-30,
-40,-20,  0,  5,  5,  0,-20,-40,
-50,-40,-30,-30,-30,-30,-40,-50,
```

Figura 8: Matriz de evaluaciones.

3. Análisis

En un principio, bajo la evaluación de perdedor/ganador que sucedía posterior a la simulación, el controlador bajo esas recompensas determinaba la jugada de nuestro equipo, lo que no nos traía resultados muy gratos, si bien podía ganarle al controlador random y combatía con el controlador básico, siempre perdía por límite de jugadas con el último, con una diferencia de 4 a 6 caballos.
Una vez implementada la evaluación final del Owervanz Controller, cambió rotundamente la toma de decisiones, ya que evaluaba todo el tablero para efectuar una acción, lo que provocó que venciera el controlador básico con 2 a 3 caballos de diferencia o que empataron, es decir, lo hizo más competitivo.