Damas

1. Autores

Camila Montserrat Alderete González
Luis Alberto Cañete Baez
Oscar Leonardo Pedrozo González

2. Resumen

Damas es un juego muy simple de jugar en comparación al ajedrez, en forma breve el juego de damas es un juego donde las reglas son que Cada jugador controla las piezas de un color situadas al comienzo a cada lado del tablero. Empieza el juego las blancas. Los movimientos se hacen por turno, uno por jugador, en diagonal, una sola casilla y en sentido de avance, o sea, hacia el campo del oponente. Si un jugador consigue llevar una de sus fichas al lado contrario del tablero cambiará dicho peón por una dama o reina (dos fichas del mismo color una encima de otra).La dama se mueve también en diagonal, pero puede hacerlo hacia delante y hacia atrás. Según las opciones de mesa puede avanzar una casilla como el peón o recorrer cualquier número de casillas mientras estén libres. Nunca podrá saltar por encima de sus propias piezas o dos piezas contiguas.Una pieza puede capturar otra si puede saltar por encima de ella siempre en dirección de ataque y en diagonal y caer en la casilla inmediatamente detrás de aquella. Finaliza la partida cuando un jugador abandona, se queda sin fichas o estas no tienen posibilidad de movimiento (bloqueo o ahogada).Se llega a tablas o empate cuando ambos jugadores lo pactan.
En este artículo se aborda el problema que supone el juego de Damas desde el punto de vista de la computación y los algoritmos como la complejidad que supone estos, donde realizan pruebas de rendimiento para realizar comparaciones de rendimiento y posteriormente dar una conclusión.

3. Introducción

El juego de Damas se cree que surgió en algún lugar entre Francia y España entre los años 1000-1500 d.C. En sus comienzos el juego se llamó *«ferses»*, nombre que ostentaba la reina en el juego de ajedrez. El movimiento de las fichas del juego de damas fue tomado de los que hacía la reina del ajedrez de acuerdo a las reglas del juego en esa época. Años más tarde a las damas se le incorporaría un nuevo movimiento que le daba la ventaja de poder saltar sobre otras fichas para realizar su captura.
Arthur Lee Samuel fue uno de los pioneros en el campo de la inteligencia artificial y los videojuegos informáticos, quien acuñó el término "aprendizaje automático" en 1959. Creó con éxito el primer programa checkers basado en el aprendizaje automático, dando una demostración temprana de los conceptos fundamentales de la inteligencia artificial.
A pesar de que las damas tienen menos jugadas que ajedrez, también posee su complejidad de implementación. En el presente trabajo desarrollaremos el algoritmo de Minimax y de Reinforcement learning con sus comparaciones respectivas y las conclusiones a las que llegamos analizando los resultados.

4. Descripción del Problema

El juego de damas se ha implementado con las siguientes reglas.

- Se juega en un tablero de 8x8 con 64 posiciones, estas alternadas entre 32 posiciones de color oscuro y 32 de color claro.
- Las piezas de color claro se encuentran en la parte inferior a la derecha de cada jugador.

- Solo las celdas oscuras están ocupadas en el tablero.
- Las fichas poseen dos colores, estos Blanco y negro para distinguir a los jugadores de manera sencilla.
- Los jugadores pueden mover una ficha a la vez, por turnos.
- Existen dos tipos de fichas, los peones y las fichas coronadas (damas).
- Los peones pueden moverse hacia adelante en diagonal, mientras que la dama puede moverse hacia atrás y adelante en diagonal. Para capturar una ficha, se debe encontrar frente (Casilla en diagonal) a la ficha seleccionada del jugador.
- En caso de ser posible la captura, el jugador tiene la opción de capturar o mover otra ficha.
- Una captura puede ser seguida por otras de manera consecutiva, siempre que la jugada sea permitida.
- El peón se convertirá en dama al alcanzar la fila más cercana al oponente, terminando con la secuencia de captura si la hubiera.
- Luego de capturar la ficha, la dama queda en la siguiente posición de la ficha capturada. Es decir no puede ir más de una celda de captura.

5. Algoritmos Implementados

Los algoritmos que implementamos en el juego son el de Minimax, Minimax con poda alfa beta y Reinforcement learning.

El algoritmo minimax es uno de los más usados para resolver este tipo de problemas porque busca que el jugador obtenga la máxima ganancia, es decir, el mejor movimiento, y este algoritmo evalúa los posibles movimientos de tu jugador y del oponente.

La poda alfa beta busca optimizar el algoritmo de minimax, descartando las evaluaciones menos óptimas sin afectar el resultado de la evaluación. Reduciendo el tiempo de evaluación y facilitando así aumentar la profundidad de la búsqueda.

Finalmente, tenemos el algoritmo de Reinforcement learning, que a diferencia de Minimax este asocia los estados posibles con una probabilidad. De esta manera, se puede saber que tan favorable es ir a una casilla. El único inconveniente es que se debe entrenar a la máquina para que aprenda las jugadas posibles más favorables.

5.1. Algoritmo Minimax

Minimax es una especie de algoritmo de retroceso que se utiliza en la toma de decisiones y la teoría del juego para encontrar el movimiento óptimo para un jugador, asumiendo que su oponente también juega de manera óptima. Es ampliamente utilizado en juegos por turnos de dos jugadores.

En Minimax los dos jugadores se denominan maximizador y minimizador. El maximizador intenta obtener la puntuación más alta posible mientras que el minimizador intenta hacer lo contrario y obtener la puntuación más baja posible.

```
function minimax(nodo, profundidad, maxPlayer)
       if depth ==0 or nodo es un nodo terminal then
       return static evaluacion del nodo

if MaxPlayer then
       maxEva= -∞
 for each nodoHijo of nodo do
       eva= minimax(nodoHijo, profunidad-1, false)
       maxEva= max(maxEva,eva)
return maxEva
```

```
else
 minEva= +∞
 for each nodoHijo of nodo do
 eva= minimax(nodoHijo, profundidad-1, true)
 minEva= min(minEva, eva)
 return minEva
```

5.2. Algoritmo Minimax con poda alfa beta

La poda alfa-beta es una técnica de optimización para el algoritmo minimax. Reduce el tiempo de cálculo. Esto nos permite buscar mucho más rápido e incluso adentrarnos en niveles más profundos en el árbol del juego. Corta ramas en el árbol del juego que no necesitan ser buscadas porque ya existe un movimiento mejor disponible. Se llama poda Alfa-Beta porque pasa 2 parámetros adicionales en la función minimax.
Alpha es el mejor valor que el maximizador puede garantizar actualmente en ese nivel o por encima.
Beta es el mejor valor que el minimizador puede garantizar actualmente en ese nivel o superior.

```
function minimax(node, depth, isMaximizingPlayer, alpha, beta):

    if node is a leaf node :
        return value of the node

    if isMaximizingPlayer :
        bestVal = -∞
        for each child node :
            value = minimax(node, depth+1, false, alpha, beta)
            bestVal = max( bestVal, value)
            alpha = max( alpha, bestVal)
            if beta <= alpha:
                break
        return bestVal

    else :
        bestVal = +∞
        for each child node :
            value = minimax(node, depth+1, true, alpha, beta)
            bestVal = min( bestVal, value)
            beta = min( beta, bestVal)
            if beta <= alpha:
                break
        return bestVal
```

5.3. Reinforcement learning

El aprendizaje por refuerzo no necesita una función de evaluación de un estado no final para obtener la utilidad, a diferencia del otro algoritmo mencionado anteriormente. Este algoritmo se basa en la recompensa, es decir si es positiva o negativa. La recompensa se propaga a la jugada anterior del jugador, cada vez que ambos realicen una secuencia de movimientos. Esto también significa que el tiempo de aprendizaje aumenta al aumentar la cantidad de acciones posibles de cada estado o si aumenta la profundidad en la que se encuentran los estados finales.
En este algoritmo alfa se representa con los valores 0 y 1, y es un parámetro que maneja la velocidad de propagación de la recompensa. Valores cercanos a 1 de alfa producen un mejor aprendizaje.

El parámetro qRate controla la razón a la que se debería explotar los nuevos estados para tener una mayor robustez ante estados desconocidos o revisitar estados que llevan a estados ganadores para reforzarlos.

```
function reinforcement-learning
	maxProb = - ∞
	q = random(0,1)
	if q <= qRate:
		for each tablerosPosibles of movimientosPosibles(tablero:
			prob = getprobabilidad(tablerosPosibles)
			if probabilidad > maxProb:
				maxProb = prob
				tableroCandidato = tablerosPosibles
		updateProbabilidad(ultimoTablero, maxProb)
	else:
		tableroCandidato = random(movimientosPosibles(tablero))
	ultimoTablero = tablero
	return tableroCandidato

function updateProbabilidad(tablero, newprob)
	prob = getprobabilidad(tablero)
	prob = prob + alfa * (probNueva - prob)
	saveProbabilidad(tablero, prob)
```

6. Resultados Experimentales

Las pruebas experimentales fueron comparación entre algoritmos, si son más propensos a ganar o a un empate, el tiempo de ejecución, y la expansión de los nodos

Los algoritmos Minimax y Poda Alfa beta en profundidad de 2 tiene mayor índice de ganar las las jugadas contra el algoritmo de “Aprendizaje por refuerzo”, la otra parte de este índice son las jugadas en que salió ganando el Aprendizaje por refuerzo en contra un Minimax y Alfa-Beta, lo que aquí se demuestra que el aprendizaje por refuerzo no fue lo suficientemente entrenado.En cuanto al tiempo de ejecución total que demora en determinar los movimientos posibles en una jugada de menor a mayor es algoritmo de aprendizaje por refuerzo seguido de Minimax y Alfa-Beta con la misma profundidad tiempos similares y con la poda se observa una reducción del tiempo significativo.En cuanto a los nodos expandidos a la misma profundidad, minimax expande mucho más nodos y aumenta cada vez más con cada nivel de profundidad que Alfa-Beta

7. Conclusión

Se comparan métodos minimax y minimax con poda alfa-beta y aprendizaje por refuerzo. Incluso con heurísticas muy simples, se han observado buenos resultados en juegos con métodos minimax. Sin embargo, los resultados del aprendizaje por refuerzo no fueron los esperados. Esto podría deberse a un error conceptual en la implementación, un mayor tiempo de entrenamiento o una elección incorrecta de los valores de los parámetros alfa, y la inspección del tablero y sus probabilidades asociadas reveló pequeñas inconsistencias al comparar los valores asignados con los intuitivos. disponible para el tablero, pero debido a la dificultad para rastrear la secuencia. También es difícil verificar la corrección del aprendizaje al producir tal resultado, ya que puede ser un valor perfectamente válido, dependiendo de la estrategia del trabajo futuro del jugador contrario, se recomienda probar un método de aprendizaje por refuerzo diferente, tal vez combinándolo. con la heurística minimax también

se pueden aplicar diferentes métodos y comparar la corrección del nodo de expansión hola con la poda alfa-beta por la cantidad de juegos ganados

8. Referencias

1. McKinley, C.: Checkers rules. *Official game rules.* https://www.officialgamerules.org/checkers (2016). Accedido el 26 de octubre de 2022.
2. Idzham, K.: Study of Artificial Intelligence into Checkers Game using HTML and JavaScript. *IOP Conference Series: Materials Science and Engineering.* https://iopscience.iop.org/article/10.1088/1757-899X/864/1/012091/pdf (2020). Accedido el 30 de octubre de 2022.
3. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall,(1995).
4. Osinski, B., Budek, K.:What is reinforcement learning? The complete guide. *Deepsense.* (2018). Accedido el 30 octubre 2022.