UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

COMPLEJIDAD ALGORÍTMICA CC184

TRABAJO PARCIAL

"Quoridor"

Docente: Canaval Sanchez, Luis Martín

Sección: WV72

Integrantes:

Altamirano Higa, Marco Alejandro - u201919054

Tarazona Chafloque, Wilmar Alonso - u20191B871

Vidal Moreno, Gabriel - u201913263

Lima, 18 septiembre de 2020

Introducción

La capacidad de que las computadoras puedan jugar videojuegos ha existido desde su creación. Esta capacidad ha crecido exponencialmente, hasta convertirse en herramientas de estudio y análisis por parte de los jugadores, que usan el potencial computacional para mejorar su juego. Incluso, hay computadoras que son mejores que los humanos en determinados juegos, como es el caso de AlphaZero o StockFish en ajedrez.

Los videojuegos son muy útiles para el estudio de áreas como el machine learning y la inteligencia artificial. Además, proporcionan un adecuado espacio para la aplicación de algoritmos en búsqueda de soluciones y la optimización de estos.

En este caso, el material de estudio para la búsqueda e implementación de algoritmos será el juego Quoridor. No es un juego muy popular, es relativamente nuevo (1997), por lo tanto no ha sido analizado exhaustivamente. Sin embargo, existen documentos donde se intenta profundizar y crear jugadores virtuales.

Quoridor consiste en un tablero de 9 filas y 9 columnas, en el que pueden participar de 2 a 4 jugadores. El jugador tiene la capacidad de desplazarse de una celda a otra hacia adelante o hacia los costados, y de poner paredes, con el propósito de obstaculizar el camino del oponente. Gana el jugador que llegue al lado opuesto antes que los demás participantes.

El proyecto se basa en la elaboración de 3 algoritmos, los cuales representarán a un bot y se enfocarán en su movimiento en el tablero, de acuerdo a las acciones del oponente. Se usará el lenguaje de programación Javascript y se desarrollará el programa en un entorno web.

Estado del arte

Continuaremos esta sección describiendo las investigaciones que ya se han realizado sobre este tema.

El primer registro de un programa capaz de jugar Quorridor corresponde a Lisa Glendenning (2005). Donde el jugador artificial utiliza un algoritmo de búsqueda “minimax” en el árbol del juego y una función de evaluación ponderada lineal en las hojas. Además, para encontrar el camino más corto para ganar utiliza el algoritmo de Dijkstra. Los resultados demuestran que no se logra un algoritmo óptimo que sea capaz de ofrecer un verdadero desafío a un jugador humano. Sin embargo, el aporte del proyecto ayuda a sentar las bases para investigaciones futuras.

Otro intento de un jugador artificial de Quoridor fue desarrollado por P. J. C. Mertens (2006). En este caso se utiliza el algoritmo de búsqueda MiniMax con “Alpha-Beta pruning”, limitando la profundidad de la búsqueda. Además, se utiliza una función de evaluación para determinar el valor de una posición con el fin de permitir un retorno más rápido. Por otro lado, para encontrar el camino más corto utiliza el algoritmo BFS (Breadth First Search). El resultado obtenido tampoco logra el objetivo de un jugador competitivo contra humanos.

Una investigación más reciente por Massagué & Brown (2018) se plantea desde las bases sentadas por los dos proyectos anteriores. Los autores optan por el algoritmo “Monte Carlo tree search” para la IA del jugador de Quorridor. Para la experimentación, evalúan el desempeño de su jugador contra otros tipos de IA de Quorridor. En los resultados se evidenció que su “agente” tiene el mejor ratio de victorias de todos los competidores. Pese a ello, los resultados no son suficientes para determinar su nivel contra humanos.

Metodología

En esta sección se explicarán las técnicas y conceptos de programación más importantes para la realización del proyecto, además del cronograma de trabajo establecido.

Para el desarrollo de la solución se utilizarán los conceptos aprendidos durante las sesiones del curso de Complejidad Algorítmica, ya que se identificaron estos temas como relevantes. Entre estos temas, los más importantes para dicha solución son: backtracking, matrices y espacios de búsqueda.

Se identificó el concepto de backtracking como uno relevante para la realización del proyecto, debido a que los jugadores de “Quoridor” tienen “el objetivo de [...] llegar hasta la base del rival” considerando que en el camino habrán obstáculos establecidos por otros rivales. Por lo tanto, esta técnica ayudará a la solución porque realizará una “búsqueda sistemática a través de todas las configuraciones posibles dentro de un espacio de búsqueda” (Jorge Baier Aranda).
En base a esta definición de backtracking, se concluyó que también se utilizarían los conceptos de espacios de búsqueda para la solución.
Finalmente, las matrices serán utilizadas para poder representar las posiciones de los jugadores y obstáculos de manera visual de forma que sea coherente para estos mismos.

A través del desarrollo de la solución, se usará la metodología ágil de desarrollo de software. De esta manera, se realizarán partes del trabajo de manera continua en forma de entregables antes de la semana 7. Seguidamente, durantes las sesiones de clases se recibirá retroalimentación de los algoritmos principales y se evaluará la eficiencia de estos mismos.

Experimentos

Algoritmos

Para el movimiento de los bots trabajaremos con los siguientes algoritmos para encontrar el mejor camino para ganar:

- Algoritmo de búsqueda A*
- Búsqueda en profundidad (DFS)
- Backtracking

Espacio de búsqueda

El espacio de búsqueda del problema consiste en la “cantidad de diferentes posibles estados” (S) que puede ocurrir en un juego. Para Quoridor esto se calcula el producto de la “cantidad de formas de ubicar un jugador” (Sj) y de la “cantidad de formas de ubicar paredes” (Sp). Apoyándonos de la investigación de Mertens (2006), planteamos las siguientes ecuaciones para calcular S:

(1) $$S_p = 81 \times 80 \times 79 \times 78 = 39929760$$

(2) $$S_f = \sum_{i=0}^{20} \prod_{j=0}^{i} (128 - 4i) = 6.1582\ 10^{38}$$

(3) $$S = S_p \times S_f = 39929760 \times 6.1582 \cdot 10^{38} = 2.4590 \cdot 10^{46}$$

Tamaños de entradas de datos para trabajar

- Lista de adyacencia de Vértices para representar el tablero de juego
- Arreglo de enteros para representar las posiciones de victoria
- Arreglo de enteros para representar las posiciones de los jugadores

Métricas de eficiencia

En la experimentación un jugador jugará contra los 3 algoritmos en 4 tableros con muros preestablecidos. Para poder identificar qué algoritmo se desempeña mejor en cada tablero se jugarán múltiples partidas en las que se intercambie las posiciones en el tablero y se intercambie los turnos de juego. Con esto podemos verificar que el algoritmo más eficiente no depende de estos factores externos (posición y turno).

Para evaluar la eficiencia de los algoritmos utilizados se tomará en cuenta las siguientes medidas:

- Cantidad de partidas ganadas
- Tiempo de ejecución
- Cantidad de errores imprevistos

Resultados

De la experimentación realizada conseguimos los siguientes resultados:

Con esto podemos interpretar que el algoritmo más eficiente es el A*, debido a que garantiza encontrar el camino más corto y actualiza este camino según la posición de los muros y la posición del jugador.

Conclusiones

En conclusión, durante la realización del trabajo parcial se vieron a mayor profundidad los temas de grafos, espacios de búsqueda, DFS y backtracking. Estos temas fueron explicados durante las sesiones de clase, sin embargo, se pudieron entender a mayor profundidad durante el proceso de programación. Además, en el desarrollo de los NPCs se tuvieron que diseñar algoritmos que pudieran razonar a partir de una entrada que iba variando según el movimiento de las demás fichas.

Bibliografía

Mertens, P.J. (2006) A quoridor-playing agent. Bachelor thesis, Department of Knowledge Engineering, Maastricht University

Glendenning, L. (2005) Mastering quoridor. Bachelor thesis, Department of Computer Science, The University of New Mexico

Massagué, Victor & Brown, Joseph (2018). Monte Carlo Tree Search for Quoridor.

Baier Aranda, Jorge (2014) Conceptos Generales de Backtracking PUC