Proyecto 2 - Connect 4 Genético

Tecnológico de Costa Rica

Ingeniería en Computación

IC6200 - Inteligencia Artificial

Cartago, II Semestre 2018

Miembros:

Kahho Chen Li

Erick Hernández López

Siul Mongalo Monge

El propósito de este proyecto es desarrollar un agente que sepa jugar Connect 4 utilizando una combinación de técnicas de búsqueda, además, entre las funcionalidades principales que logra el proyecto es entrenar agentes que puedan jugar Connect 4 interactivamente con un humano, y tener a la vez la posibilidad de establecer una sesión de juego humano vs máquina o máquina vs máquina, a través de la interfaz de consola.

Algoritmo General del juego

Esta sección del proyecto se implementa a través de una clase “Tablero”, la cual contiene diferentes métodos para hacer funciones de manejo del objeto tablero, con el propósito de realizar los procesos en el momento del análisis de cada jugada. Esta clase tiene un constructor que cuenta con atributos de cantidad de filas y cantidad de columnas, para lo requerido en este proyecto la totalidad de filas es 6, mientras que la de columnas es 7, esto con el fin de cumplir con los criterios del juego.

Para esta sección, se implementan métodos como cambiar valores en el tableros, obtener filas y columnas de diferentes maneras, analizar la secuencia de las piezas para determinar futuros ganes, entre otras que hacen restricciones de componentes para obtener el análisis de juego buscado.

Algoritmo Agente

Este algoritmo es el que determina muchas decisiones en base a los cálculos que se ejecutan, ya que éste cuenta con los atributos de las diferentes estrategias implementadas para la selección de campos en el tablero. Esta clase además es la encargada de ejecutar las diferentes acciones para modificar el ambiente del juego, en la cual hay métodos que cuentan con las funciones de detectar ganes, bloquear gane de oponentes o ejecutar su gane actual, incluyendo la llamada a la función de costo implementada a través de una clase del algoritmo MiniMax frecuente en los juegos.

Estrategias de juego

Se plantean cuatro estrategias básicas para la selección de la jugada del agente cuando no debe de bloquear un gane o bien ejecutar su propio gane. Se realiza la programación de las dos estrategias propuestas en el enunciado secuencia vs espacios y centros vs extremos, además de las mismas se proponen dos estrategias más según la investigación realizada sobre estrategias de juego para connect4, las mismas son fila vs. columna y pares vs. impares. El desarrollo de estas estrategias se realiza en la clase agente en donde el constructor asigna una probabilidad a cada estrategia para ser seleccionada por el mismo. A continuación se detalla un poco más el funcionamiento de cada una de las estrategias:

Centro vs. Extremos

Esta estrategia retorna las posiciones de centros cuando la estrategia busca favorecer la colocación de fichas en las columnas centrales del tablero y en caso contrario retorna las posiciones de los extremos extremos según la probabilidad generada por el random.

Secuencia vs. Espacios

Esta estrategia buscar retornar las posiciones consecutivas a una ficha ya colocada buscando generar secuencias para lo cual realiza una búsqueda de las fichas ya colocadas o bien en caso contrario busca dejar espacios entre fichas para futuras jugadas de gane.

Fila vs. Columna

Esta estrategia lleva una idea similar a la secuencia pero buscando favorecer diferentes direcciones; en el caso de filas da prioridad a jugadas que puedan generar secuencias de cuatro fichas en fila (horizontal) y en el caso de columna da prioridad a jugadas que puedan generar secuencias de cuatro fichas en columna (vertical).

Pares vs. Impares

Según la investigación algunas estrategias buscan favorecer la colocación de fichas siempre en posiciones impares por lo tanto esta estrategia tiene por objetivo retornar las posiciones de columnas pares o impares según la estrategia seleccionada aleatoriamente

Algoritmo MiniMax

Este algoritmo es el encargado de hacer la selección de la columna para ejecutar el mejor movimiento buscando cumplir con los criterios del juego, ya sea ganar o evitar que el oponente gane. Esta clase al instanciarse recibe el tablero del juego actual y el vector de estrategias seleccionado por las probabilidades anteriormente explicadas. Es importante mencionar que al momento de ejecutar el método que hace la búsqueda de ese campo, éste recibe el jugador actual y el tablero para ser duplicado, con el propósito de evitar modificaciones en el tablero principal.

El objetivo de este clase es buscar la mejor opción basada en la consecutividad de las fichas presentes hasta el momento, por lo cual, la asignación de los pesos a cada nodo del árbol generado en la recursividad es dependiente a la cantidad de fichas que haya alrededor de la analizada. Para ello, se consideran las cuatro modalidades en la que se puede obtener una victoria, las cuales son por medio de cuatro fichas seguidas verticalmente, horizontalmente, diagonal con pendiente positiva o diagonal con pendiente negativa; para cada una de las anteriores se le asigna un peso de 100000 si hay posibilidad de poner la cuarta ficha consecutiva, un peso de 100 si existe la posibilidad de poner la tercera ficha consecutiva y la suma de la cantidad de veces en las cuales se les puede poner la segunda ficha consecutiva con el peso de 1.

Algoritmo Genético

Para la optimización de las características seleccionadas anteriormente, se utilizará un algoritmo genético. En este caso la población inicial es un arreglo de agentes inicializados con valores aleatorios para la cual se deberán enfrentar todos contra todos acumulando victorias. Esto resulta ser la función de “fitness” en la cual, los agentes con las características más importantes prevalecerá mediante la cantidad de victorias que puedan generar.

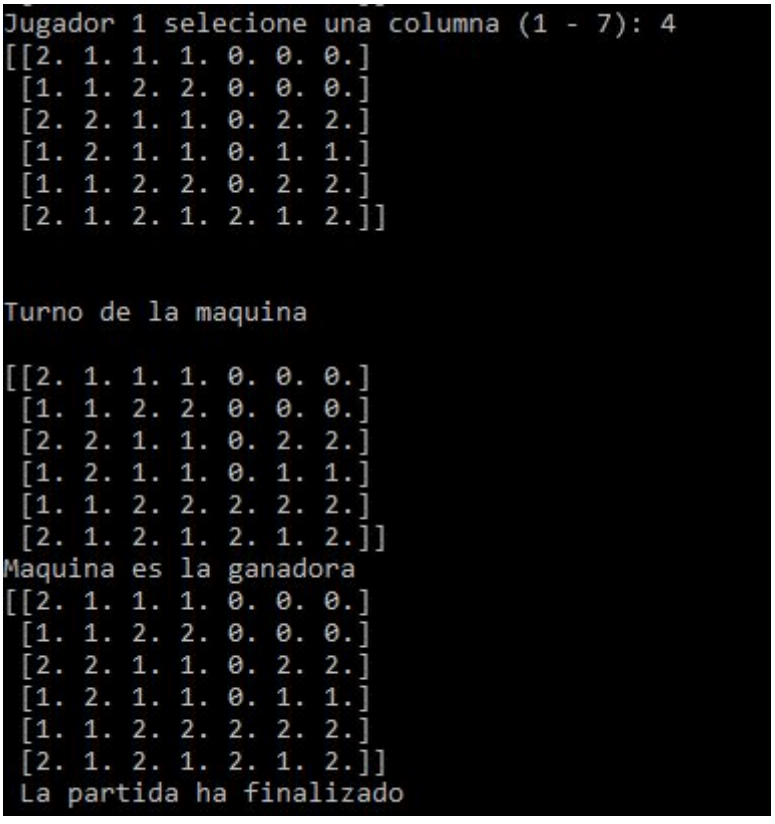
Se podrá configurar el algoritmo genético mediante parámetros en la consola. Estas configuraciones son:

- Cantidad de Agentes: el número de agentes que se conservan de generación en generación, estos agentes son los que tienen el mejor valor de “fitness”.
- Cantidad de Generaciones: el número de generaciones que los agentes deberán “entrenar” para llegar a su mejor configuración.
- Tamaño de población: cantidad de agentes en cada generación. Es la población inicial y debe mantenerse esa cantidad a lo largo de las generaciones.

Además de que los agentes deben jugar entre ellos, se realizará operaciones de cruce y mutaciones. De cada generación, los agentes con los mejores valores de “fitness” no solo son preservados, sino que su configuración será cruzada con otro agente para generar un nuevo agente. En este caso, la estrategia que se implementó es usar los mejores pares de agentes y cruzar su configuración. También se genera 1 sola mutación por generación, la estrategia implementada es seleccionar el mejor agente y realizar un cambio en solo una característica de su configuración y así crear un nuevo agente “mutado”.

Casos ejecutados según modalidad de juego

Ejemplo Humano vs. Máquina

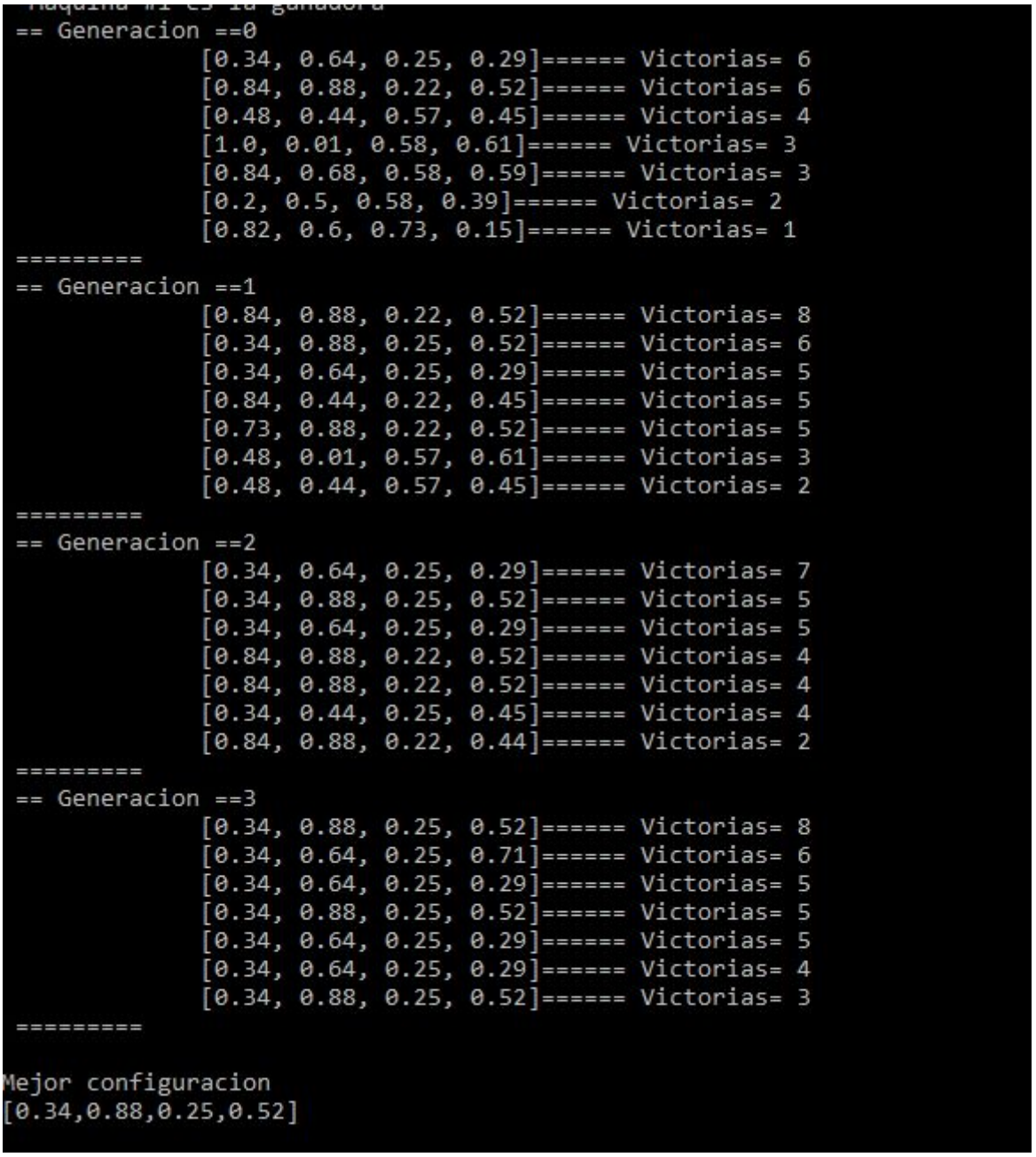


Ejemplo Máquina vs. Máquina

```
== Generacion ==0
                  [0.73, 0.79, 0.8, 0.12]====== Victorias= 4
                  [0.81, 0.17, 0.16, 0.82]====== Victorias= 3
                  [0.47, 0.13, 0.97, 0.26]====== Victorias= 3
                  [0.83, 0.07, 0.81, 0.45]====== Victorias= 1
                  [0.36, 0.68, 0.55, 0.85]====== Victorias= 1
=========
== Generacion ==1
                  [0.81, 0.17, 0.16, 0.82]====== Victorias= 8
                  [0.73, 0.79, 0.8, 0.12]====== Victorias= 5
                  [0.73, 0.17, 0.8, 0.82]====== Victorias= 3
                  [0.73, 0.79, 0.59, 0.12]====== Victorias= 2
                  [0.81, 0.13, 0.16, 0.26]====== Victorias= 1
=========
== Generacion ==2
                  [0.81, 0.17, 0.16, 0.82]====== Victorias= 11
                  [0.73, 0.79, 0.8, 0.12]====== Victorias= 8
                  [0.73, 0.17, 0.8, 0.82]====== Victorias= 4
                  [0.73, 0.79, 0.8, 0.41]====== Victorias= 3
                  [0.81, 0.79, 0.16, 0.12]====== Victorias= 2
=========
== Generacion ==3
                  [0.81, 0.17, 0.16, 0.82]====== Victorias= 11
                  [0.73, 0.79, 0.8, 0.12]====== Victorias= 10
                  [0.73, 0.38, 0.8, 0.12]====== Victorias= 5
                  [0.73, 0.17, 0.8, 0.82]====== Victorias= 3
                  [0.81, 0.79, 0.16, 0.12]====== Victorias= 1
=========

Mejor configuracion
[0.81,0.17,0.16,0.82]
```

Para esta corrida de máquina vs máquina se guardó de forma acumulada las victorias de cada agente.



Para las siguientes corridas de cada generación se maneja de forma independiente cada victoria.