Informe de Contribución Individual - Reto Connect-4

Universidad de La Sabana | Facultad de Ingeniería Asignatura: Fundamentos de Inteligencia Artificial
Estudiante: Giovanni Moreno ("Gio") **Fecha:** 24 de Noviembre de 2025

1. Resumen de Contribución

Mi rol principal en el proyecto fue el diseño y la implementación de la arquitectura híbrida del agente (`GioImprovedPolicy`). Ante las limitaciones de tiempo de cómputo y la necesidad de un rendimiento robusto desde la primera partida, decidí descartar un enfoque puro de Monte Carlo Tree Search (MCTS) en favor de una solución que integra **Búsqueda Adversaria (Minimax)** con **Aprendizaje por Refuerzo Tabular**.

Me enfoqué en dotar al agente de una "intuición base" mediante heurísticas posicionales para evitar el problema de arranque en frío (Cold Start) típico de los agentes de RL, asegurando así el cumplimiento del requisito de vencer al jugador aleatorio desde la semana 1.

2. Aportes Destacados (Highlights)

A continuación, presento mis dos contribuciones técnicas más significativas:

Aporte A: Implementación del Motor Híbrido (Minimax + Q-Learning)

Diseñé la lógica central en `act` y `minimax` para que el agente tome decisiones basándose en una búsqueda de profundidad limitada (`depth=4`) con Poda Alfa-Beta. La innovación clave fue integrar la **Q-Table** como función de evaluación en los nodos hoja: si el estado es conocido, el agente usa su "memoria" (RL); si es desconocido, utiliza la heurística. Esto permite validación empírica del aprendizaje sin sacrificar competencia inmediata.

- **Archivo:** `groups/C/policy.py`
- **Evidencia (Commit):** 🔗 Ver Commit: Integración final Híbrida Q-Learning + Minimax

Aporte B: Diseño de Heurística Posicional y Optimización de Poda

Para optimizar la velocidad del algoritmo Minimax, implementé una heurística matemática (`score_position`) que valora el control del centro y las ventanas de 4 fichas. Adicionalmente, implementé una lógica de ordenamiento de acciones (`sorted(valid_cols, key=lambda x: abs(x - 3))`) que prioriza la evaluación de columnas centrales. Esto maximiza la ocurrencia de podas Alfa-Beta, reduciendo drásticamente el tiempo de cómputo y permitiendo una búsqueda más profunda.

- **Archivo:** `groups/C/policy.py`
- **Evidencia (Commit):** 🔗 Ver Commit: Heurística posicional y Center Control

3. Desafíos y Logros

Desafíos Enfrentados:

- **Latencia en Python:** La implementación inicial de Minimax era lenta. El desafío fue optimizar la poda. Descubrí que al evaluar primero las jugadas centrales (estadísticamente mejores), el algoritmo descarta ramas inútiles mucho más rápido.
- **Gestión de Memoria:** Un MCTS puro reseteaba el conocimiento en cada turno. El reto fue estructurar la persistencia de datos para que el agente pudiera "recordar" entre partidas (usando `pickle`), integrando esto dentro de la estructura de carpetas del torneo (`groups/C`).

Principales Logros:

- **Invencibilidad vs. Random:** Se logró un agente que gana el 100% de las partidas contra un jugador aleatorio, cumpliendo el requisito crítico de la primera entrega.
- **Defensa Reactiva:** Implementación de un sistema de "bloqueo de emergencia" que detecta victorias inminentes del rival antes de entrar a la búsqueda profunda, evitando derrotas por "ceguera de horizonte".

---

4. Reflexión y Propuestas de Mejora

Reflexión de la Solución: La solución híbrida demostró ser superior a las heurísticas estáticas y al MCTS con pocas simulaciones. La capacidad de usar una Q-Table permite que el agente evolucione, mientras que la heurística garantiza que nunca juegue de forma "tonta" en estados nuevos. Sin embargo, la dependencia de una tabla hash limita la generalización: el agente no entiende que dos estados son similares, solo sabe si son idénticos.

Propuestas de Mejora:

1. **Uso de Bitboards:** Reemplazar la matriz de Numpy por operaciones a nivel de bits (bitwise operations). Esto permitiría evaluar el tablero en nanosegundos y aumentar la profundidad de búsqueda de 4 a 8 o más.
2. **Aproximación de Funciones (Deep RL):** Sustituir la Q-Table tabular por una pequeña Red Neuronal. Esto permitiría al agente generalizar patrones visuales y estimar el valor de estados nunca vistos, reduciendo el uso de memoria RAM.