Sistemas Multiagente

Obligatorio 2

Integrantes:
Martin Rizzo - 343631
Leandro Cardoso - 166267

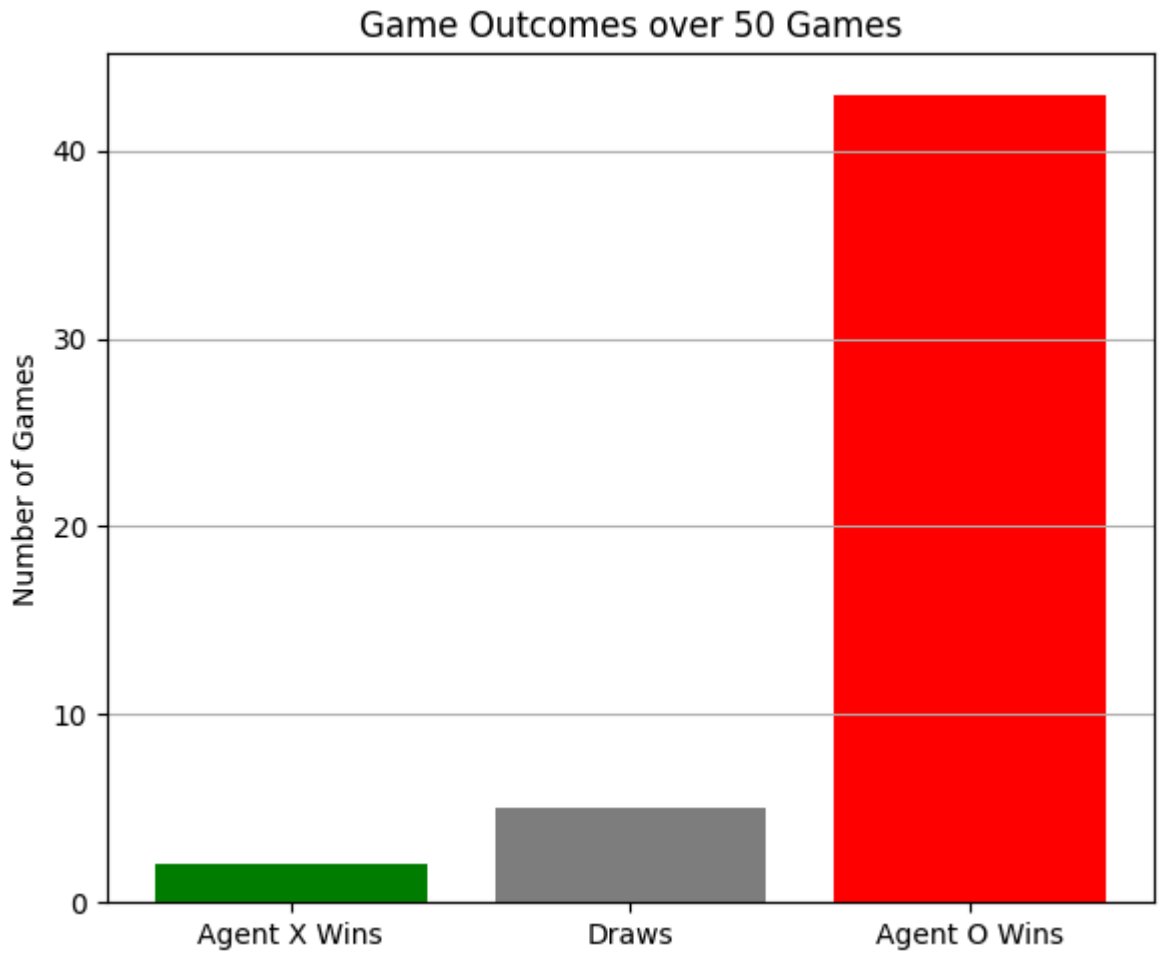
Introducción

El siguiente informe procura realizar un análisis sobre dos agentes en una serie de ambientes de juegos alternados. Los agentes a ser analizados son: Monte Carlo Tree Search (MCTS), Minimax y Counterfactual Regret Minimization (CFRM). Los agentes serán utilizados en los ambientes: Tic-Tac-Toe, Nocca-Nocca y Kuhn para 2 y 3 jugadores.
A diferencia de los juegos simultáneos, los ambientes de juego alternado se caracterizan por que cada jugador realiza una acción luego de que su contrincante haya realizado una acción.

Tic-Tac-Toe

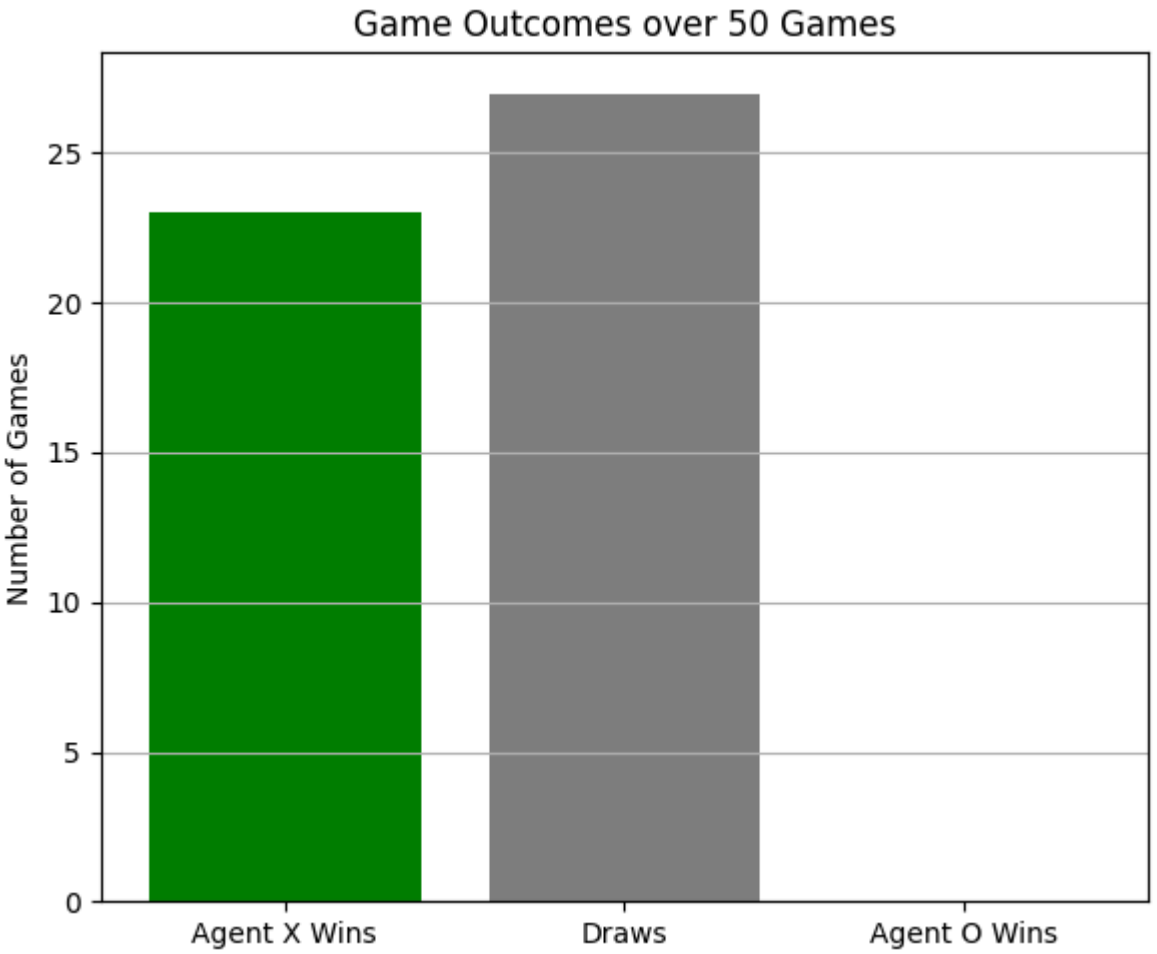
El ambiente representa el clásico juego del ta-te-ti. Donde existen dos jugadores y cada uno procura hacer tres en línea con su elemento, sea una cruz (X) o un círculo (O). Dicho juego es el más básico del que se dispone y sirve en gran parte para verificar la correcta implementación de los agentes.

RANDOM VS MCTS

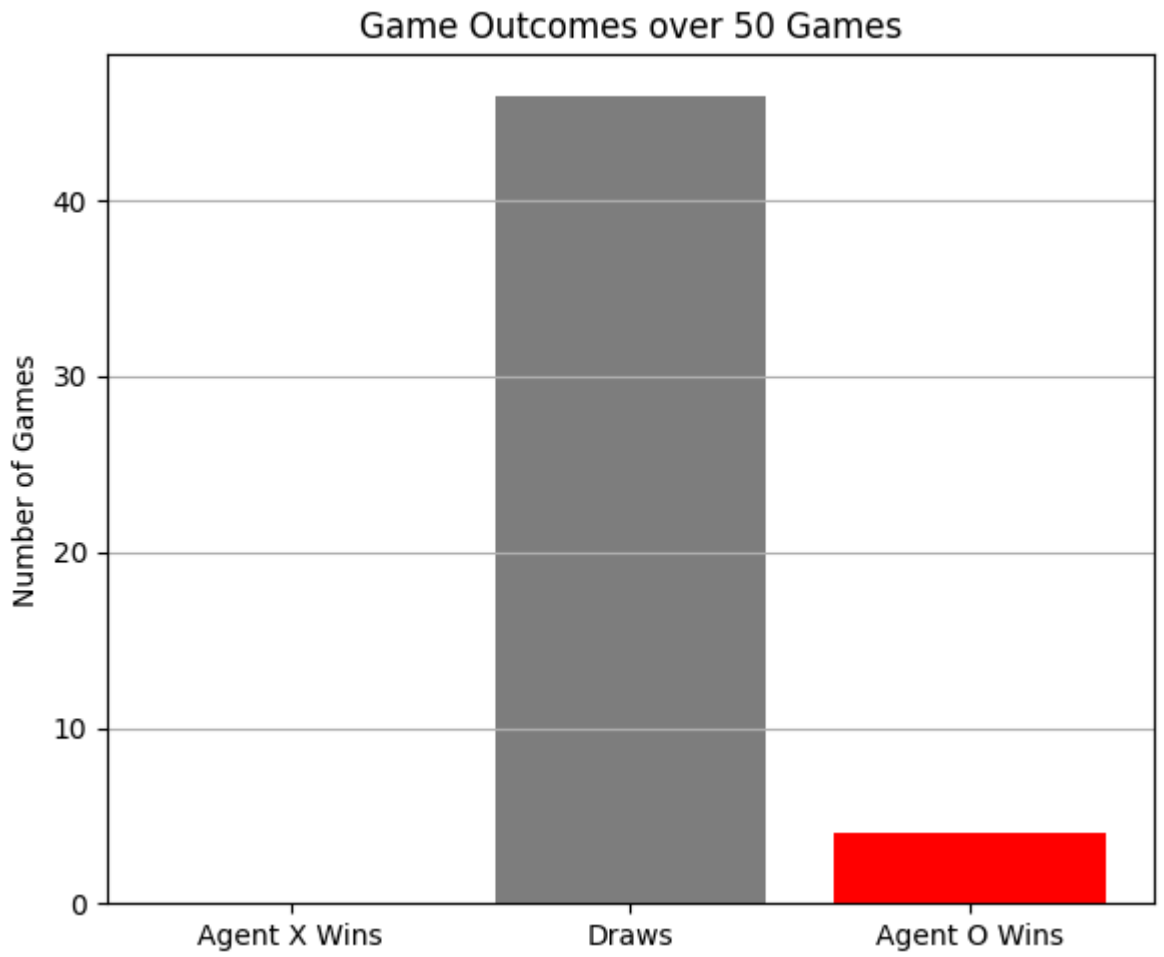


MINIMAX VS MCTS

A continuacion se muestra un experimento con un agente MiniMax de profundidad 3 (X) y un MCTS con 150 simulaciones y 10 rollouts (O). Se observa que MiniMax no pierde nunca y el peor resultado es un empate.



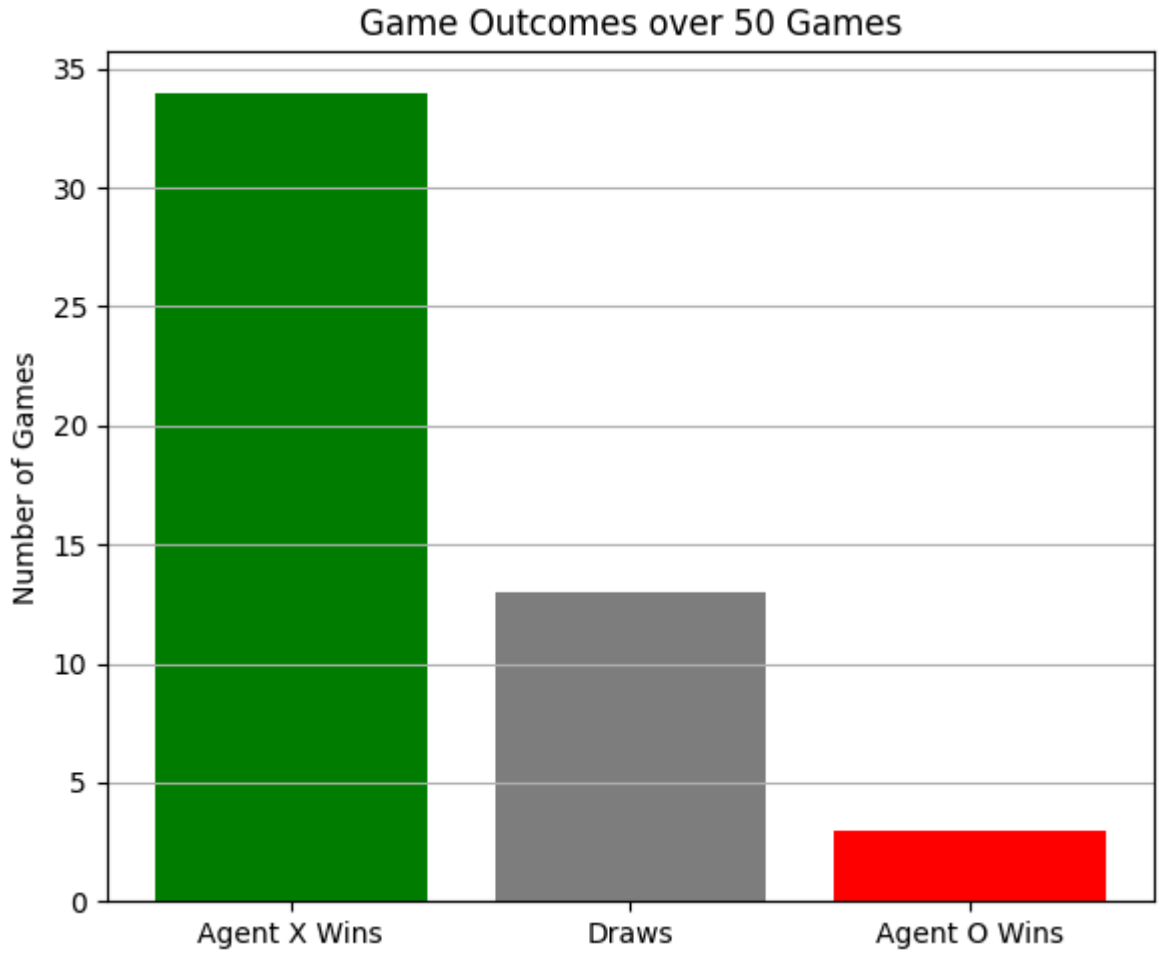
Incluso si se juega el primer movimiento con MCTS y se baja la profundidad de MiniMax a 2, el primero nunca gana. Se producen mayormente empates y en ocasiones gana MiniMax aun no siendo el que comienza la partida.



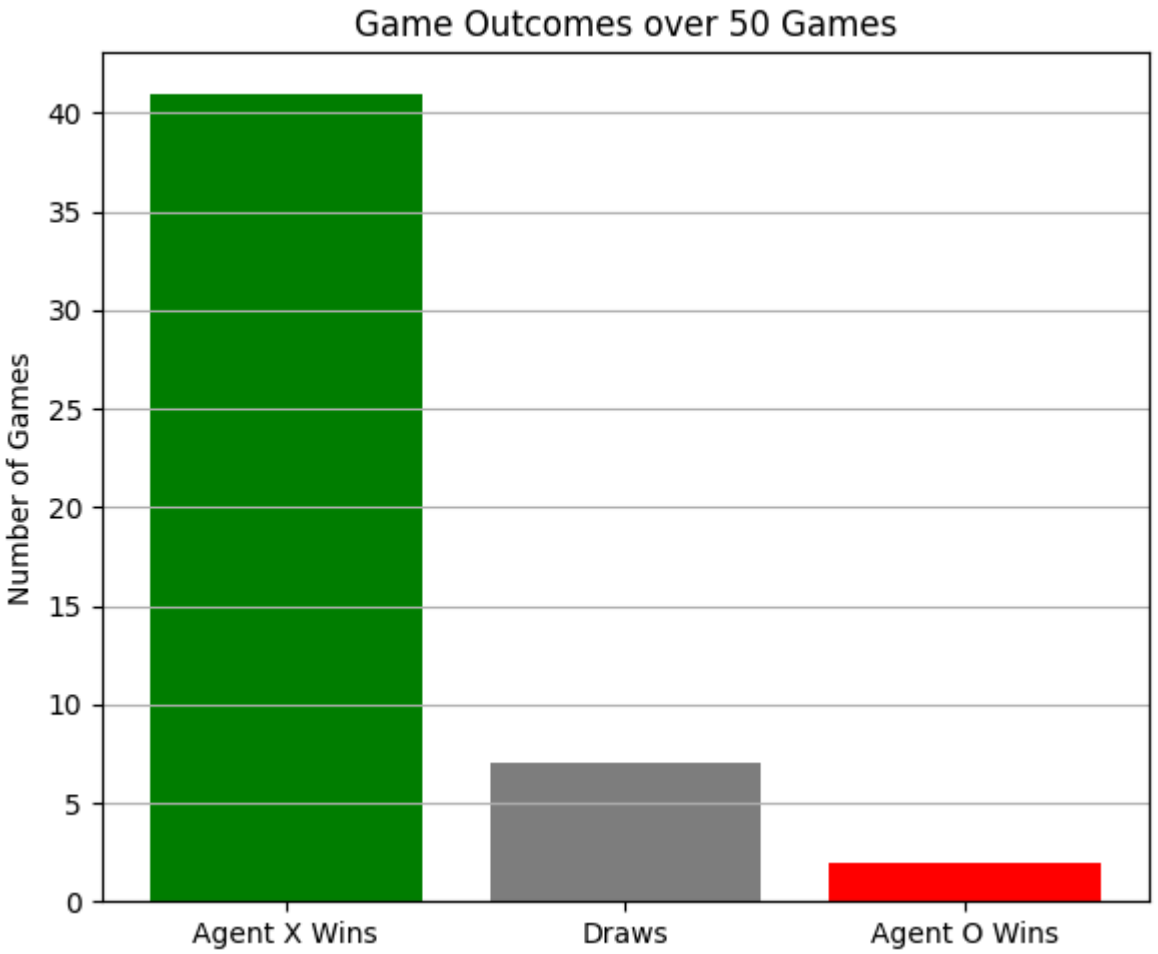
MCTS VS MCTS

Al correr dos agentes MCTS en Tic Tac Toe, un agente con 100 simulaciones (X) y el otro con 150 (O) y la misma cantidad de rollouts (10), se puede observar que la mayoría de las partidas las gana el agente X, a pesar de tener menos simulaciones. Esto se debe a que en este juego en particular, el agente que hace el primer movimiento tiene ventaja. De todas formas se produce una cantidad de empates no despreciable y una menor cantidad de

victorias del agente O.



Si se corren dos MCTS exactamente iguales, el efecto es más notable. En este caso se utilizaron dos MCTS con 100 simulaciones y se ven aún menos empates, favoreciendo aun más al jugador que comienza la partida.



Nocca-Nocca

Es el juego más complejo de los tres en cuanto a la profundidad del árbol y la cantidad de acciones. Consiste en un tablero de 5x8 donde cada jugador (negro y blanco) comienza el juego con 5 fichas en las filas 1 y 6 respectivamente. Los movimientos son de una posición en cualquier dirección (similar al rey en el ajedrez). Se pueden superponer hasta 3 fichas en una misma posición formando una pila y para ganar uno de los jugadores tiene que conseguir llegar hasta el extremo opuesto del tablero (fila 0 si el jugador arranca con sus fichas en la fila 6 o bien fila 7 para el jugador cuyas fichas comienzan en la fila 1), o dejar una ficha apilada en cada una de las fichas del oponente.

Función de evaluación

Para este juego se implementó una función de evaluación de tres sumandos para el agente MCTS.

Uno de los términos consiste en el avance promedio de las piezas. Para cada jugador, la función calcula un promedio normalizado utilizando la posición de cada pieza de dichos jugadores. En un caso la función será mayor cuando las piezas se acerquen en promedio a

la fila 7 (por ejemplo: agente Black) y en el otro caso será mayor mientras más cerca estén en promedio a la fila 0 (por ejemplo: agente White), y luego se convierten para que quepan en el rango [-1,1], para que cuando todas las piezas estén lo más atrás posible el resultado de este término sea -1 y cuando estén todas avanzadas sea 1.

Otro término tiene en cuenta las fichas bloqueadas por el oponente y las que están libres de moverse. Consiste en hallar la proporción normalizada de piezas bloqueadas sobre el total de las piezas para cada jugador, y luego un reescalado al intervalo [-1,1]. De este modo mientras más piezas bloqueadas, más se acerca a -1 este término, y por el contrario mientras más fichas libres se acercara más a 1.

El último término representa la “amenaza de victoria”. Si hay una pieza del oponente en la fila donde comienza el agente (una ficha del agente White en la fila 1 o una ficha del agente Black en la fila 6) y devuelve 1 si dicha condición se cumple o 0 si esto no ocurre.

Finalmente los tres términos se ponderan según la importancia que se les quiere dar. En particular se le dio más importancia al avance promedio, luego a las fichas bloqueadas y por último la amenaza de victoria.

MCTS vs Random Agent

Se realizó un experimento de Random Agent (Black) contra un agente MCTS (White) sin función de evaluación y otro utilizando dicha función utilizando 4 rollouts. En ambos casos le gana consistentemente a Random Agent, lo cual es de esperar ya que Nocca Nocca es un juego con mucha mayor complejidad que Tic Tac Toe donde es muy poco probable que jugando movimientos al azar se le pueda ganar a un agente como MCTS. Para esta parte se muestra como resultado de los experimentos la captura del tablero en la última jugada indicando al agente ganador y el turno en el que gana.

Turn 33 -- Agent White plays action 64
0: ___ ___ ___ 1__ ___
1: ___ ___ ___ 0__ 00_
2: 0__ 0__ 1__ ___ ___
3: ___ ___ ___ ___ ___
4: 1__ ___ ___ ___ ___
5: ___ ___ ___ ___ ___
6: ___ ___ ___ 1__ 1__
7: ___ ___ ___ ___ ___
Reward agent Black: -1
Reward agent White: 1
The winner is: White

Turn 29 -- Agent White plays action 40
0: 1__ ___ ___ ___ ___
1: ___ ___ ___ ___ ___
2: 0__ 00_ ___ 0__ ___
3: ___ ___ ___ 0__ ___
4: ___ ___ ___ ___ ___
5: ___ ___ 1__ ___ 1__
6: ___ ___ ___ 11_ ___
7: ___ ___ ___ ___ ___
Reward agent Black: -1
Reward agent White: 1
The winner is: White

MCTS vs MCTS con eval

Se realizaron experimentos entre dos agentes MCTS, donde uno utiliza la función de evaluación implementada, y el otro no. Ambos con la misma cantidad de rollouts y simulaciones. En general tiende a ganar el que no utiliza la función de evaluación, lo cual tiene sentido ya que recorre una mayor parte del árbol para tomar decisiones, aunque ciertas veces puede ganar el que utiliza la función implementada. También se midieron los tiempos de selección de la acción para ambos agentes, y son muy similares.

El agente White corresponde en todos los casos al MCTS con la función de evaluación implementada.

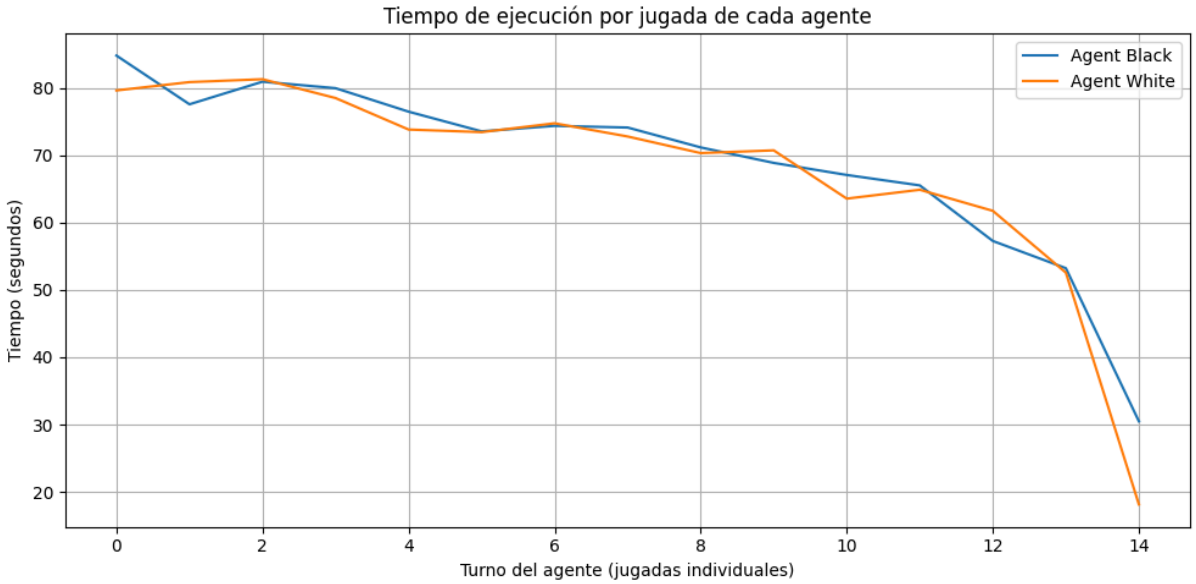
Los siguientes resultados de experimentos utilizan 100 simulaciones y 10 rollouts para ambos agentes.

Turn 32 -- Agent Black plays action 273
0: ___ ___ ___ ___ ___
1: ___ 0__ 0__ ___ ___
2: ___ ___ ___ 00_ ___
3: ___ ___ ___ 1__ ___
4: ___ ___ ___ 11_ ___
5: ___ ___ ___ 1__ 1__
6: ___ ___ ___ ___ ___
7: ___ ___ ___ ___ 0__
Reward agent Black: 1
Reward agent White: -1
The winner is: Black

Turn 29 -- Agent White plays action 72
0: ___ ___ ___ ___ 1__
1: ___ 00_ ___ ___ 0__
2: ___ ___ ___ 0__ ___

3: ___ ___ 0__ ___ ___
4: ___ 1__ ___ ___ ___
5: 1__ ___ ___ ___ ___
6: 1__ ___ 1__ ___ ___
7: ___ ___ ___ ___ ___
Reward agent Black: -1
Reward agent White: 1
The winner is: White

Turn 40 -- Agent Black plays action 257
0: ___ ___ ___ ___ ___
1: ___ 0__ ___ 0__ ___
2: 0__ 1__ 1__ ___ ___
3: ___ ___ ___ 0__ ___
4: ___ ___ ___ ___ ___
5: ___ 1__ ___ ___ ___
6: ___ ___ ___ 1__ 1__
7: ___ ___ 0__ ___ ___
Reward agent Black: 1
Reward agent White: -1
The winner is: Black



Al reducir la cantidad de rollouts, se tienen ejecuciones más rápidas y menor tiempo de elección de las acciones para ambos agentes. Sin embargo, lo que demora en elegir una acción no es muy distinto al agente que no utiliza dicha función.

Turn 41 -- Agent White plays action 56
0: ___ ___ 1__ ___ ___
1: ___ 0__ ___ 0__ ___
2: ___ 00_ ___ ___ ___

3: ___ 1__ ___ ___ ___
4: ___ ___ ___ ___ ___
5: ___ ___ ___ 1__ ___
6: 1__ ___ 0__ 1__ ___
7: ___ ___ ___ ___ ___
Reward agent Black: -1
Reward agent White: 1
The winner is: White

Turn 56 -- Agent Black plays action 257
0: ___ ___ ___ ___ ___
1: 0__ ___ ___ 1__ ___
2: ___ ___ ___ 10_ ___
3: ___ ___ ___ 0__ ___
4: ___ ___ ___ 0__ ___
5: ___ 1__ ___ 1__ ___
6: ___ ___ ___ ___ 1__
7: ___ ___ 0__ ___ ___
Reward agent Black: 1
Reward agent White: -1
The winner is: Black

Turn 55 -- Agent White plays action 48
0: ___ 1__ ___ ___ ___
1: ___ 0__ ___ ___ ___
2: ___ ___ ___ ___ 0__
3: ___ ___ ___ ___ ___
4: ___ 0__ ___ 10_ ___
5: ___ ___ ___ 1__ 1__
6: ___ ___ 0__ 1__ ___
7: ___ ___ ___ ___ ___
Reward agent Black: -1
Reward agent White: 1
The winner is: White

Una posible explicación es que la función de evaluación es costosa. Ya que debe recorrer las fichas en el tablero y calcular promedios.

MCTS vs MiniMax

Se realizaron experimentos entre MiniMax con profundidad 4 y MCTS con 10 rollouts y 100 simulaciones. Una vez con función de evaluación para MCTS y otra sin esta función. En ambos casos se obtuvo un resultado similar, siendo MiniMax el que siempre gana. Las partidas entre dichos agentes son muy largas ya que Nocca Nocca es un juego con mucha profundidad en su arbol.

En el siguiente resultado se muestra MiniMax como agente Black y MCTS como agente White.

Turn 36 -- Agent Black plays action 252
0: ___ ___ ___ ___ ___
1: ___ 010 ___ ___ ___
2: ___ 0__ ___ ___ 0__
3: ___ ___ ___ ___ ___
4: ___ ___ ___ 1__ ___
5: 1__ ___ ___ 1__ ___
6: 1__ ___ ___ ___ ___
7: 0__ ___ ___ ___ ___
Reward agent Black: 1
Reward agent White: -1
The winner is: Black

A diferencia del experimento anterior, en este caso Black es MCTS y White es MiniMax.

Turn 49 -- Agent White plays action 47
0: ___ 1__ ___ ___ ___
1: ___ ___ ___ ___ 0__
2: ___ ___ 0__ ___ ___
3: ___ ___ ___ ___ ___
4: ___ ___ ___ ___ ___
5: ___ ___ ___ 01_ ___
6: ___ 101 ___ 01_ ___
7: ___ ___ ___ ___ ___
Reward agent Black: -1
Reward agent White: 1
The winner is: White

Luego se realizó un experimento bajando la profundidad a 1 para el agente MiniMax (Black). En este caso se puede observar que si MiniMax tiene poca profundidad, el agente es miope y no logra ganarle a MCTS con 10 rollouts.

Turn 23 -- Agent White plays action 56
0: ___ ___ 1__ ___ ___
1: 0__ 0__ ___ ___ 0__
2: ___ ___ ___ 0__ ___
3: ___ ___ ___ ___ ___
4: ___ 0__ ___ ___ ___
5: 1__ ___ 1__ ___ ___
6: ___ ___ 1__ ___ 1__
7: ___ ___ ___ ___ ___
Reward agent Black: -1
Reward agent White: 1
The winner is: White

Póker de Kuhn

Es un juego que simplifica al tradicional juego de poker, creado por Harold Kuhn. Esta versión cuenta con tres cartas: una jota, una reina y un rey. El juego comienza al repartir una carta para cada jugador para que luego estos puedan realizar dos únicas acciones: apostar o pasar. El jugador con la carta de mayor valor gana la partida, a menos que una apuesta no sea igualada, en cuyo caso gana quien la haya realizado.

Este tipo de juego es un entorno más complejo para los agentes de inteligencia artificial, ya que introduce estrategias claves del poker como el bluff, o mentir en cuanto a su capacidad de apostar. Por este motivo, los jugadores deben tener la capacidad de tomar decisiones sobre lo observado y posibles engaños. El Póker de Kuhn es un excelente entorno de juegos de suma cero complejos con información imperfecta.

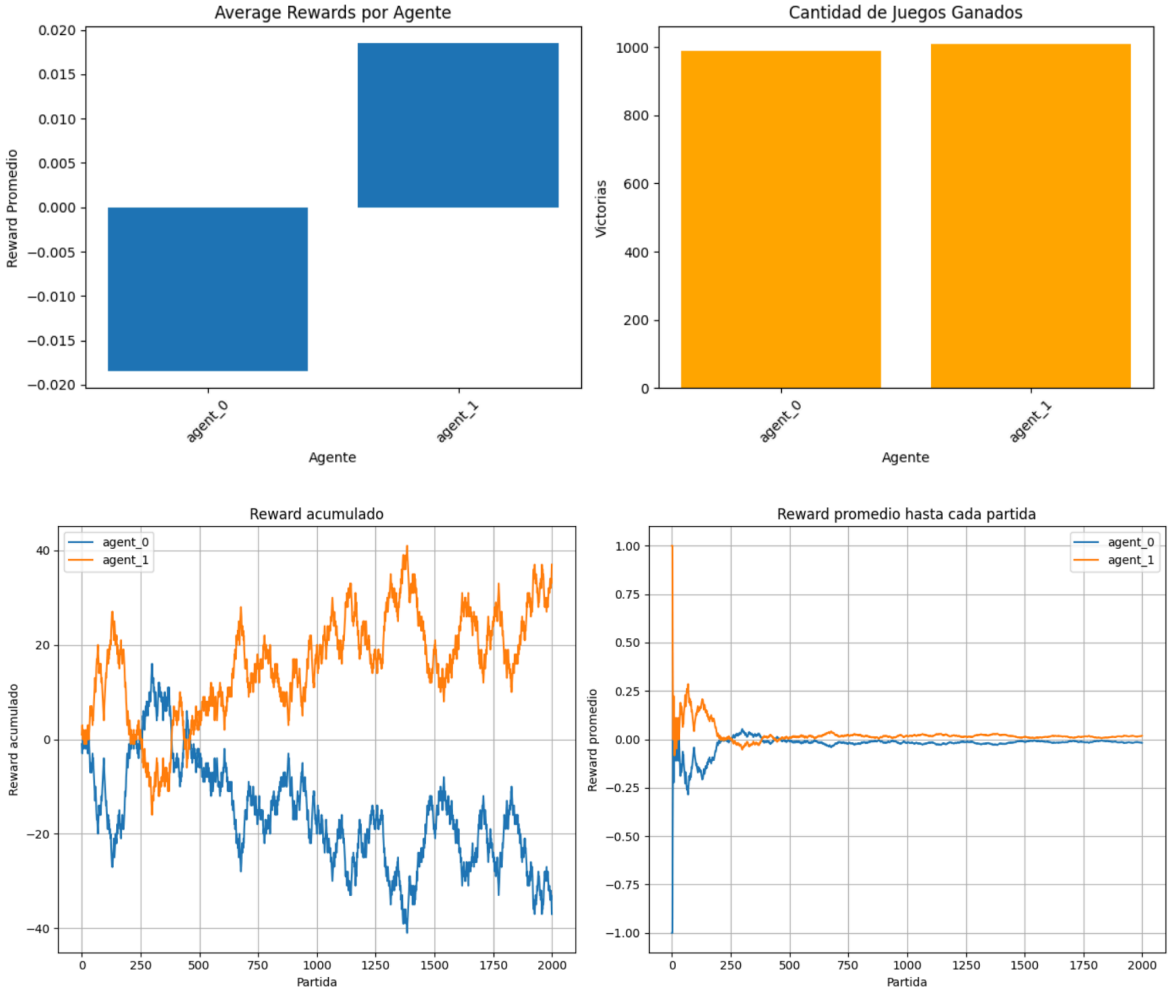
Para nuestros experimentos utilizaremos las versiones del Póker de Kuhn para dos jugadores y tres jugadores.

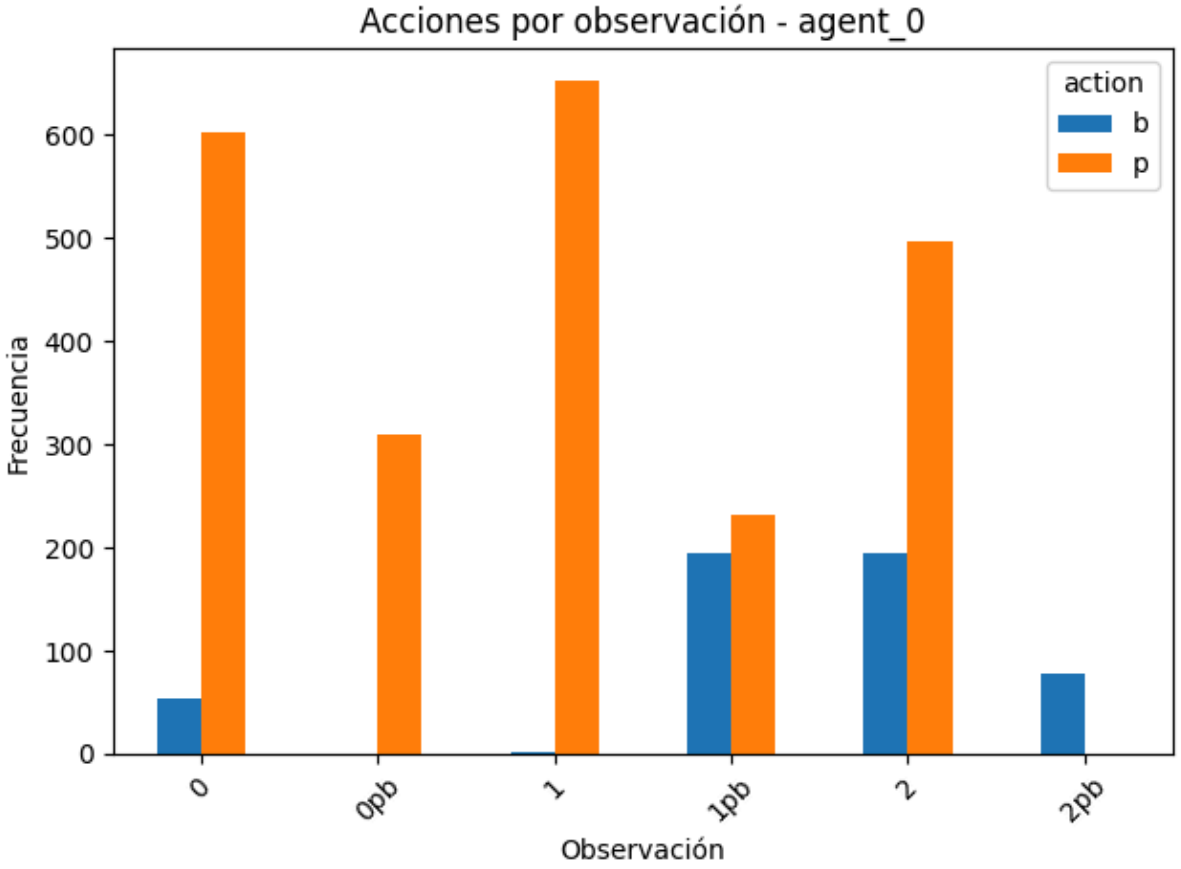
Póker de Kuhn para dos jugadores

Para la versión para dos jugadores realizaremos experimentos entre todos los agentes disponibles para observar el comportamiento de cada escenario.

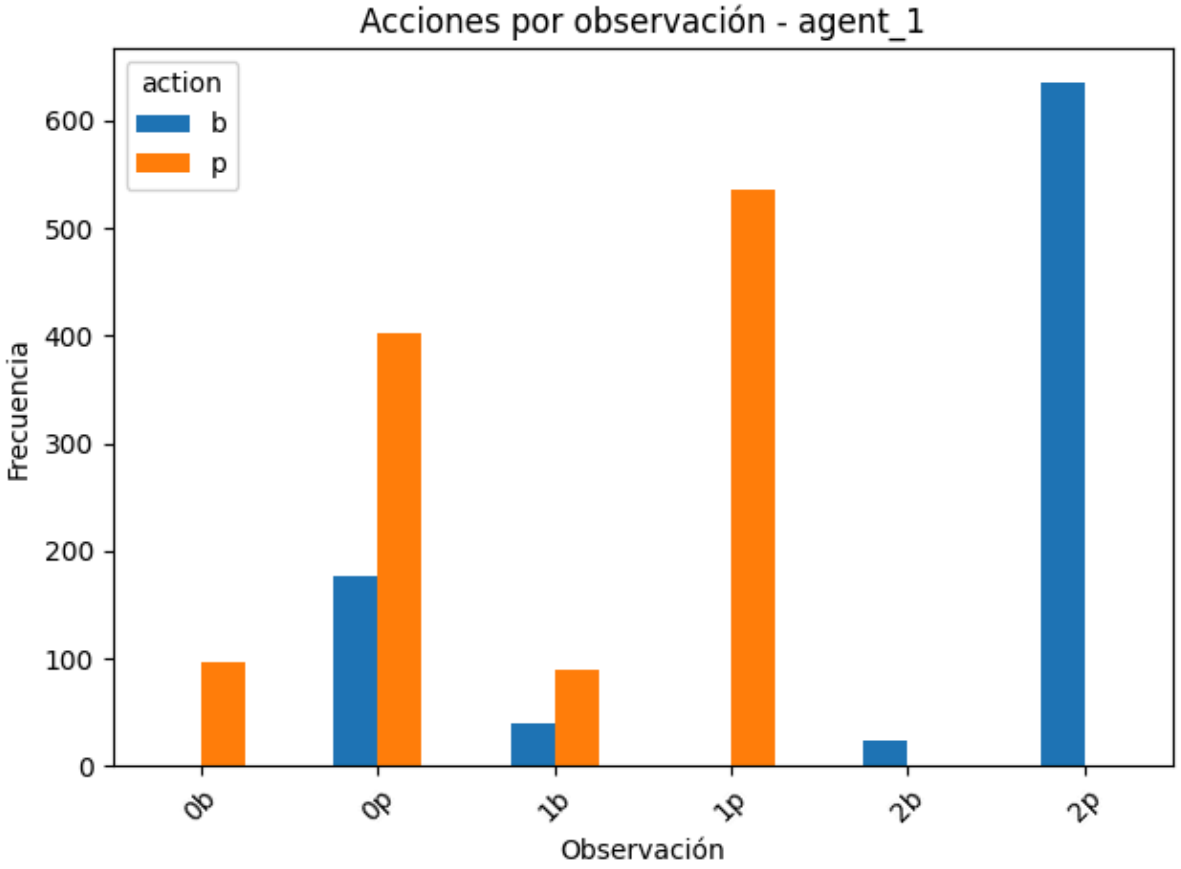
CFRM VS CFRM

El primer cruce que haremos serán dos CFRM. Al ser un juego de suma cero, se espera que exista un ganador. Lo que se observa al realizar este tipo de enfrentamiento es: que existe un equilibrio en el tiempo sobre las recompensas acumuladas y que un jugador gana sobre el otro por el hecho de tener una posición ventajosa durante el juego. Esto es pues, el agente_1 cuenta con mayor información al ser siempre el segundo agente a tomar una acción. La posición de cada agente durante cada experimento es fundamental.





	agent	observation	action	count
0	agent_0	0	b	54
1	agent_0	0	p	602
2	agent_0	0pb	p	309
3	agent_0	1	b	1
4	agent_0	1	p	652
5	agent_0	1pb	b	194
6	agent_0	1pb	p	232
7	agent_0	2	b	194
8	agent_0	2	p	497
9	agent_0	2pb	b	78

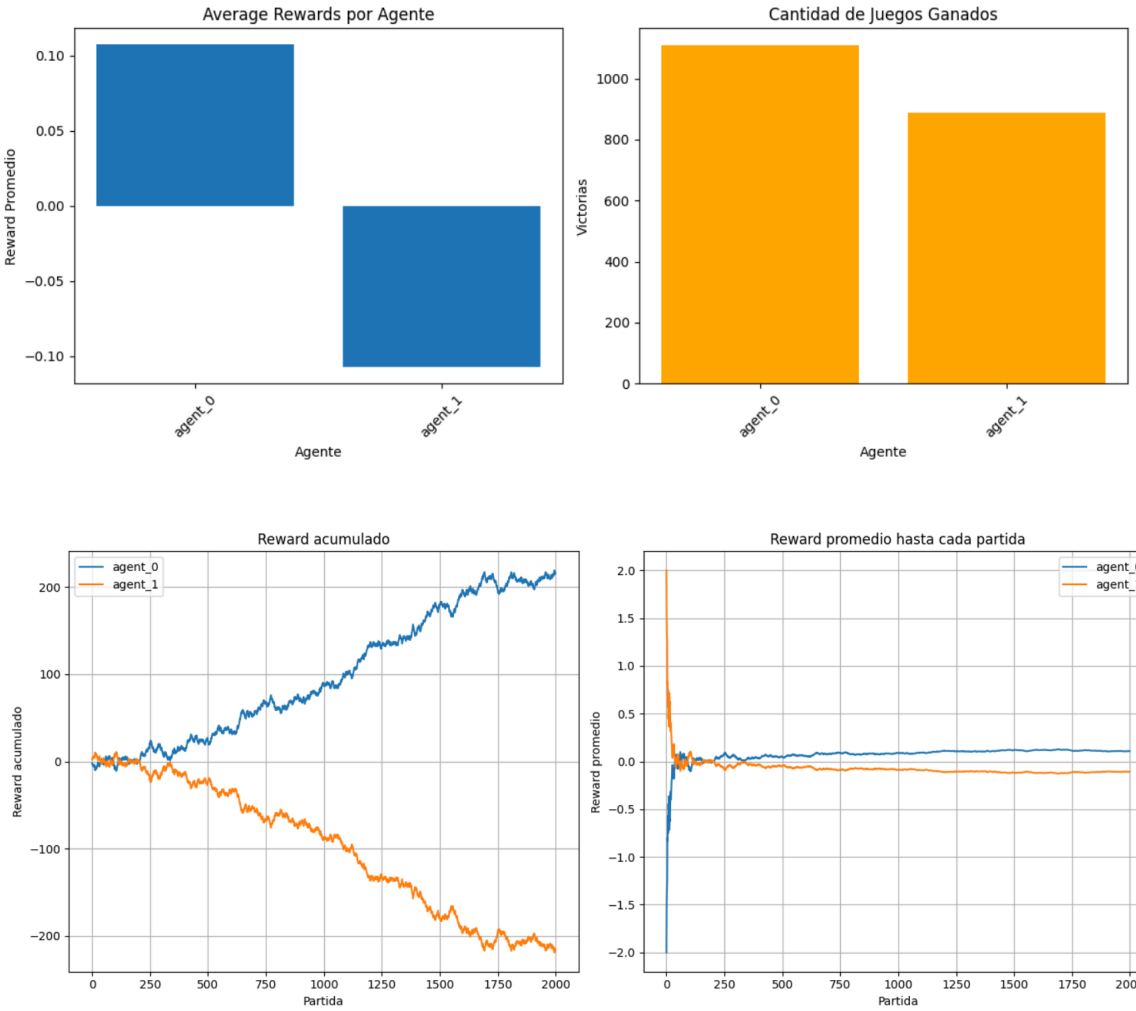


	agent	observation	action	count
10	agent_1	0b	p	96
11	agent_1	0p	b	176
12	agent_1	0p	p	402
13	agent_1	1b	b	40
14	agent_1	1b	p	90
15	agent_1	1p	b	1
16	agent_1	1p	p	536
17	agent_1	2b	b	23
18	agent_1	2p	b	636

Como puede observarse en estos últimos gráficos, el segundo agente cuenta con una ventaja comparativa al observar, por ejemplo: 2p. Es decir, cuenta con mayor información del entorno.

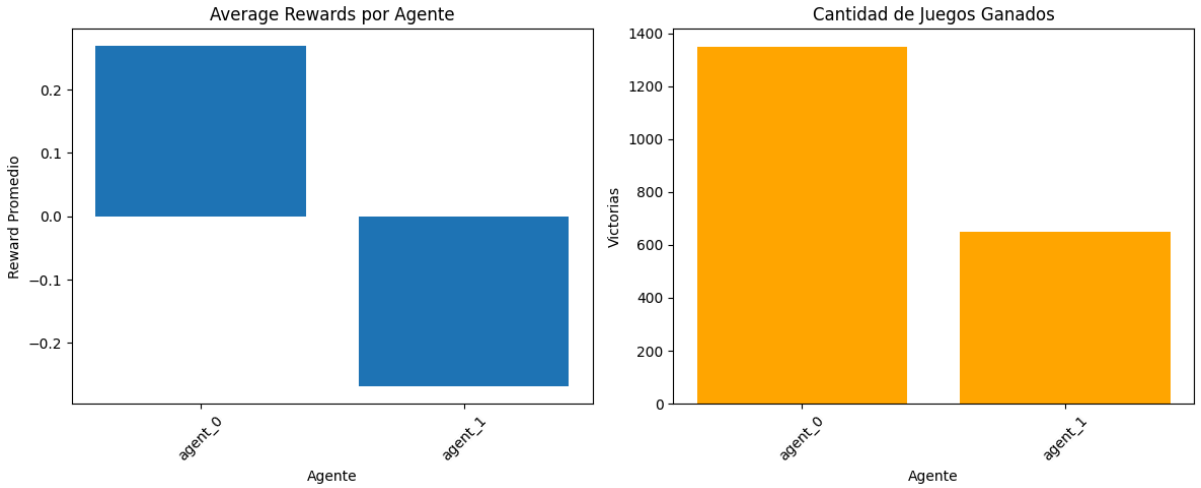
MCTS VS Random

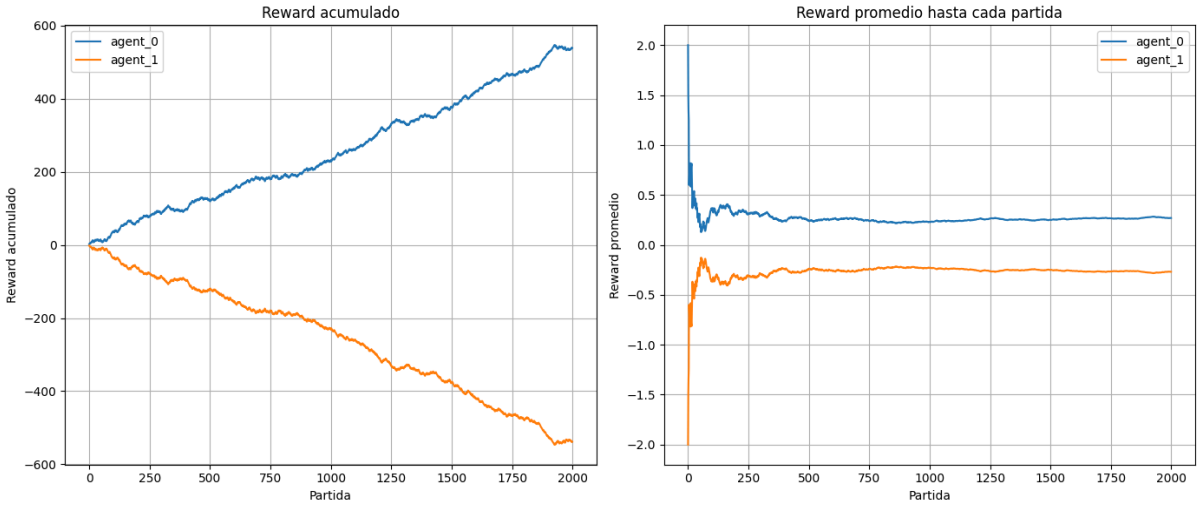
En nuestro segundo experimento enfrentamos a un agente Monte Carlo Tree Search (MCTS) con rollouts = 2 contra un agente Random. En esta configuración inicial, `agente_0` corresponde a MCTS y `agente_1` al agente Random. Observamos que con solo dos rollouts, MCTS es capaz de superar consistentemente al agente Random a partir de la partida número 125 aproximadamente. Es decir, incluso con simulaciones limitadas, el agente aprende rápidamente a explotar la debilidad del oponente.



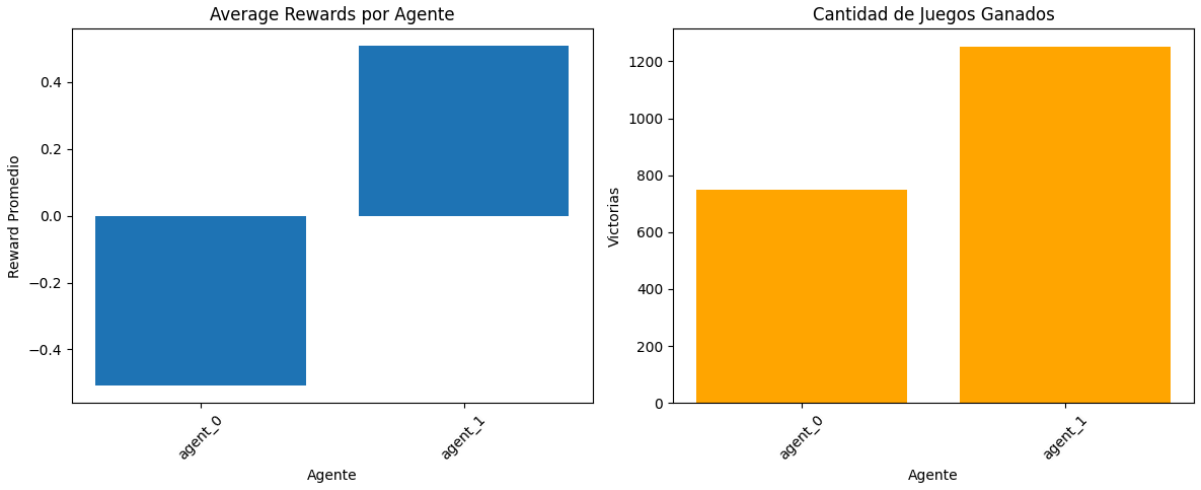
No obstante, al invertir las posiciones (es decir, `agente_0`: Random y `agente_1`: MCTS), los resultados se modifican significativamente.

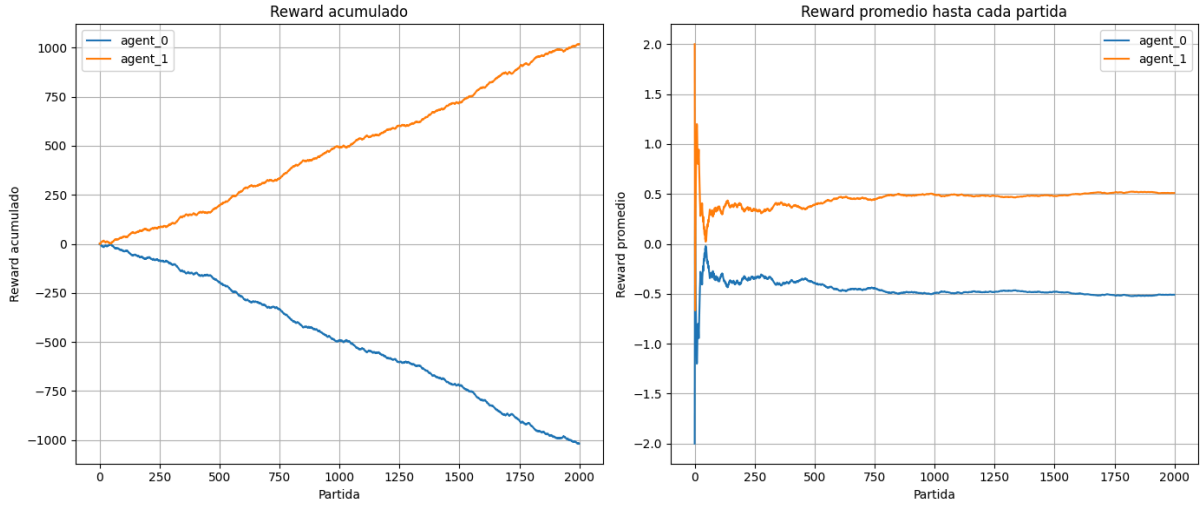
En este caso, MCTS muestra un desempeño inferior, vemos que el orden de turno impacta fuertemente el resultado. En un juego como Kuhn Poker, el jugador que inicia suele tener ventaja. Dado que el agente Random introduce comportamientos caóticos, el MCTS en segunda posición no logra establecer una estrategia ganadora con tan pocos rollouts. La incertidumbre generada por el Random, sumada a una capacidad limitada de simulación, impide que MCTS se adapte eficazmente.





Cuando los rollouts = 5, el comportamiento cambia. Con esta configuración y MCTS en segunda posición logra superar consistentemente al agente Random. Como se observa en los gráficos siguientes, a partir de este punto el agente logra una política efectiva pese a iniciar en desventaja.





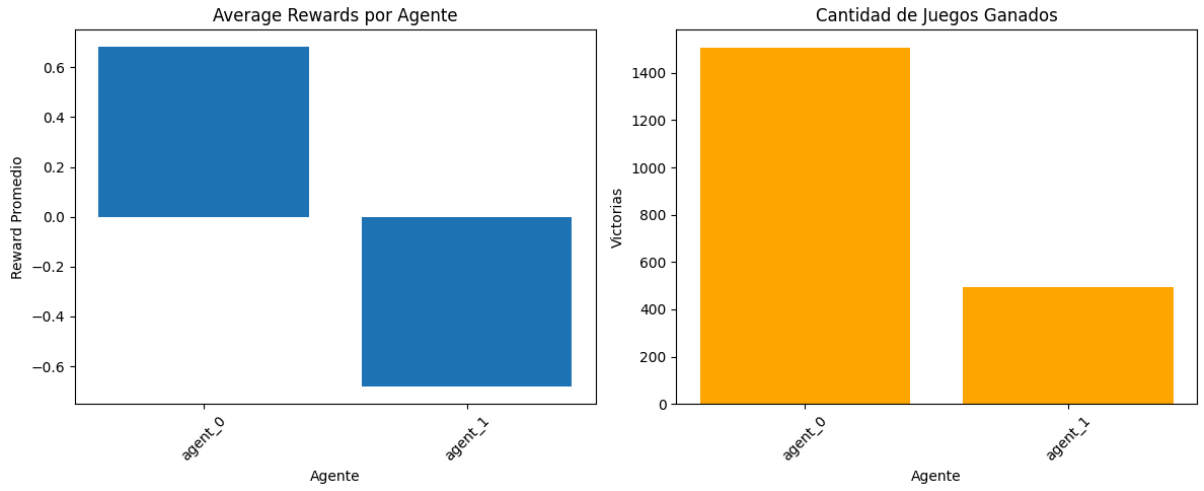
Con una mayor capacidad de simulación de posibles futuros (es decir, más rollouts), MCTS se vuelve efectivo incluso frente a oponentes impredecibles. Sin embargo, este aumento de profundidad tiene un costo computacional significativo. Por ejemplo, ejecutar MCTS con 5 rollouts durante 2000 iteraciones llevó un tiempo total de 46 minutos y 29 segundos.

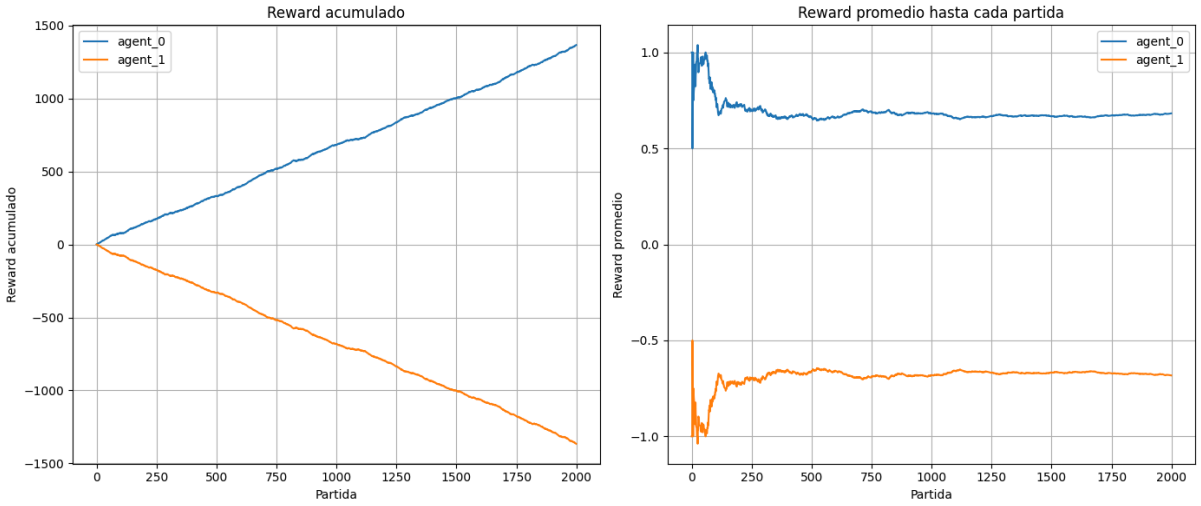
En contextos más complejos, este tipo de enfoque puede volverse muy costoso si no se dispone de suficiente tiempo o recursos. Por lo tanto, puede ser preferible optar por algoritmos más eficientes en inferencia.

CFRM VS Random

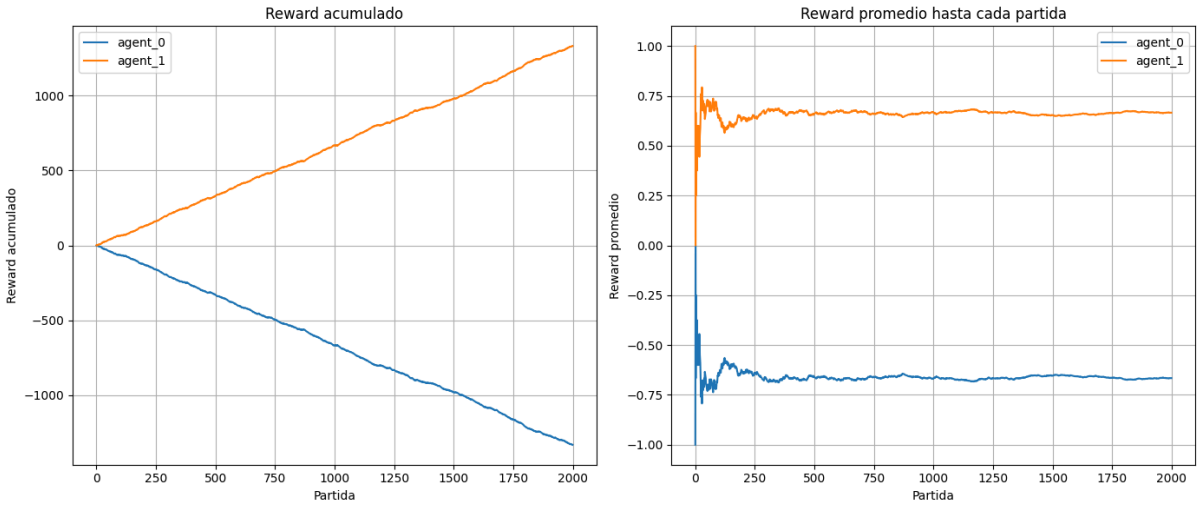
Counterfactual Regret Matching (CFRM) supera a MCTS en entornos como el Póker de Kuhn. De forma breve, CFRM basa su toma de decisión en minimizar el arrepentimiento de haber tomado una u otra acción.

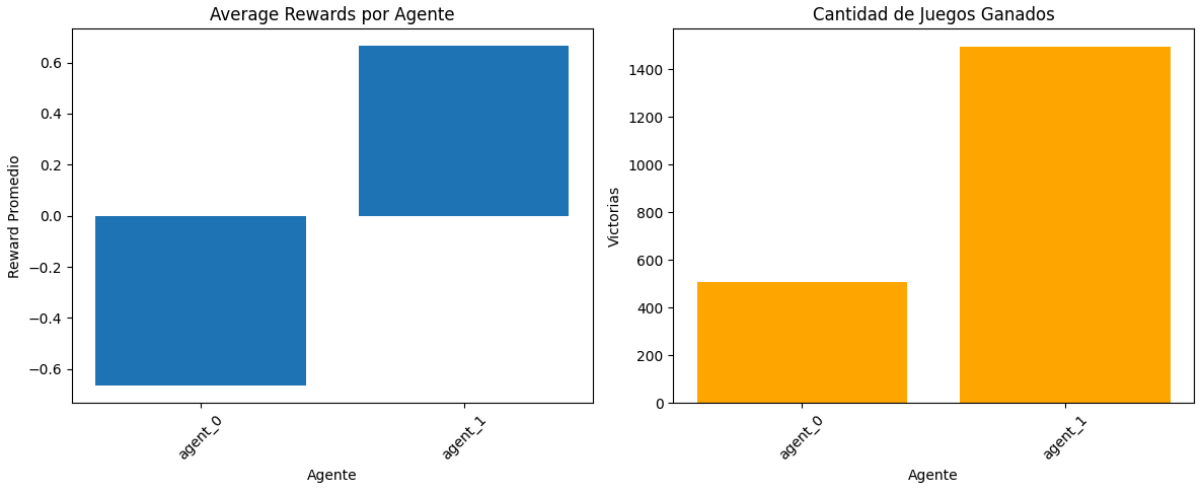
Como vemos en los siguientes gráficos, CFRM rápidamente consigue superar a un agente Random. El siguiente gráfico muestra `agente_0`: CFRM y `agente_1`: Random.





Al cambiar la posición de CFRM, éste tampoco se ve afectado. El desempeño del algoritmo no depende del orden del juego. Esto podemos apreciarlo en los siguientes gráficos (`agente_0`: Random y `agente_1`: CFRM):



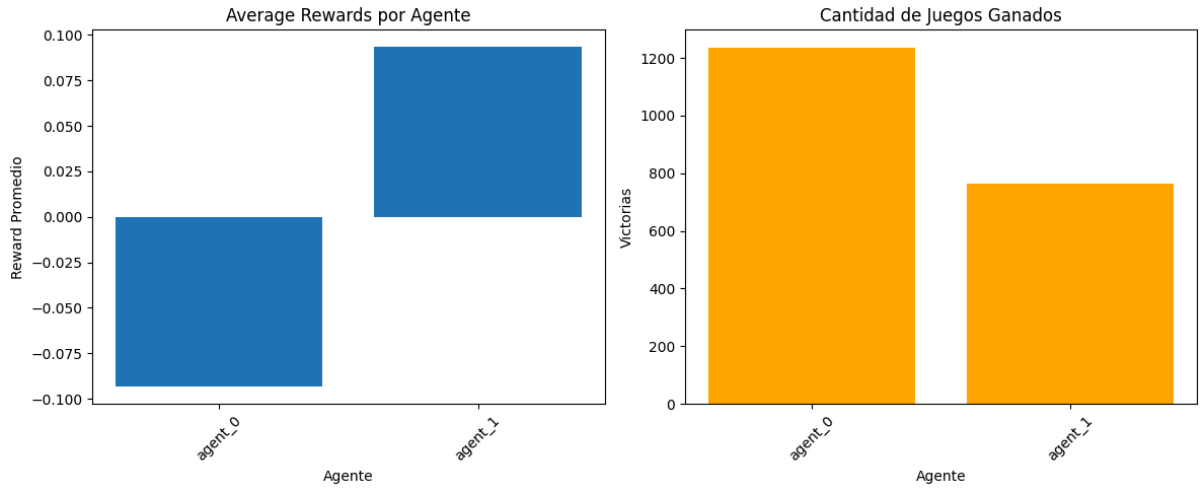


El procesamiento de estos problemas es considerablemente más eficaz por parte de CFRM. La inferencia toma apenas segundos para poder terminar 2000 iteraciones de juegos contra un agente Random. El único contratiempo de CFRM es la necesidad de realizar una serie de episodios de entrenamientos previo a comenzar a jugar contra su oponente. Para nuestra prueba, realizamos 10000 iteraciones para entrenar a nuestro agente CFRM. No obstante, el tiempo total sigue siendo ampliamente menor a comparación de MCTS en cualquiera de sus variantes.

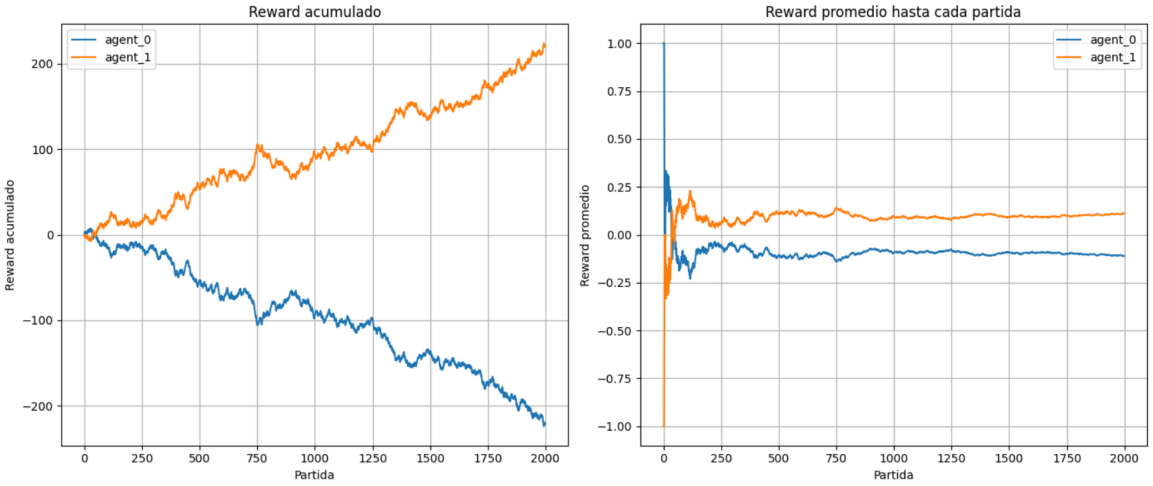
MCTS VS CFRM

Ahora bien, ¿qué sucede cuando enfrentamos a MCTS y CFRM? ¿Es verdad que necesariamente CFRM siempre será superior a MCTS? En principio la respuesta sería que sí. No obstante, al experimentar vemos algunos comportamientos interesantes.

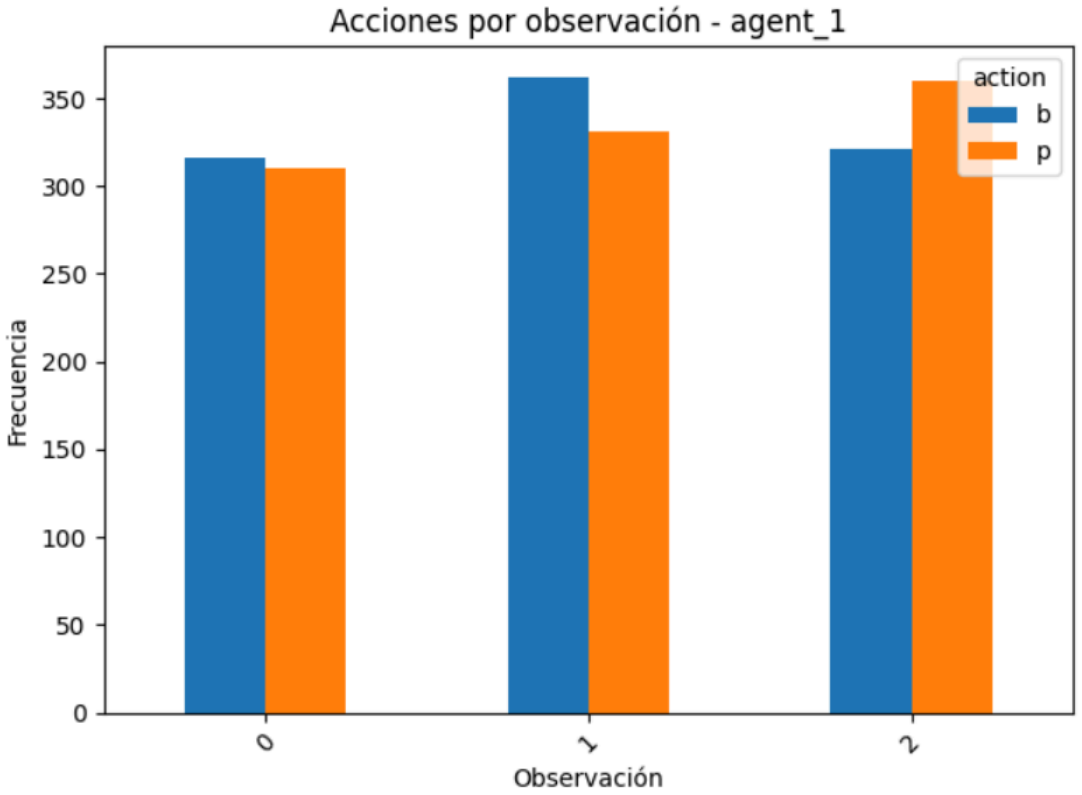
Si contamos con `agente_0`: MCTS con rollout = 5 y con `agente_1`: CFRM. Vemos el siguiente comportamiento:



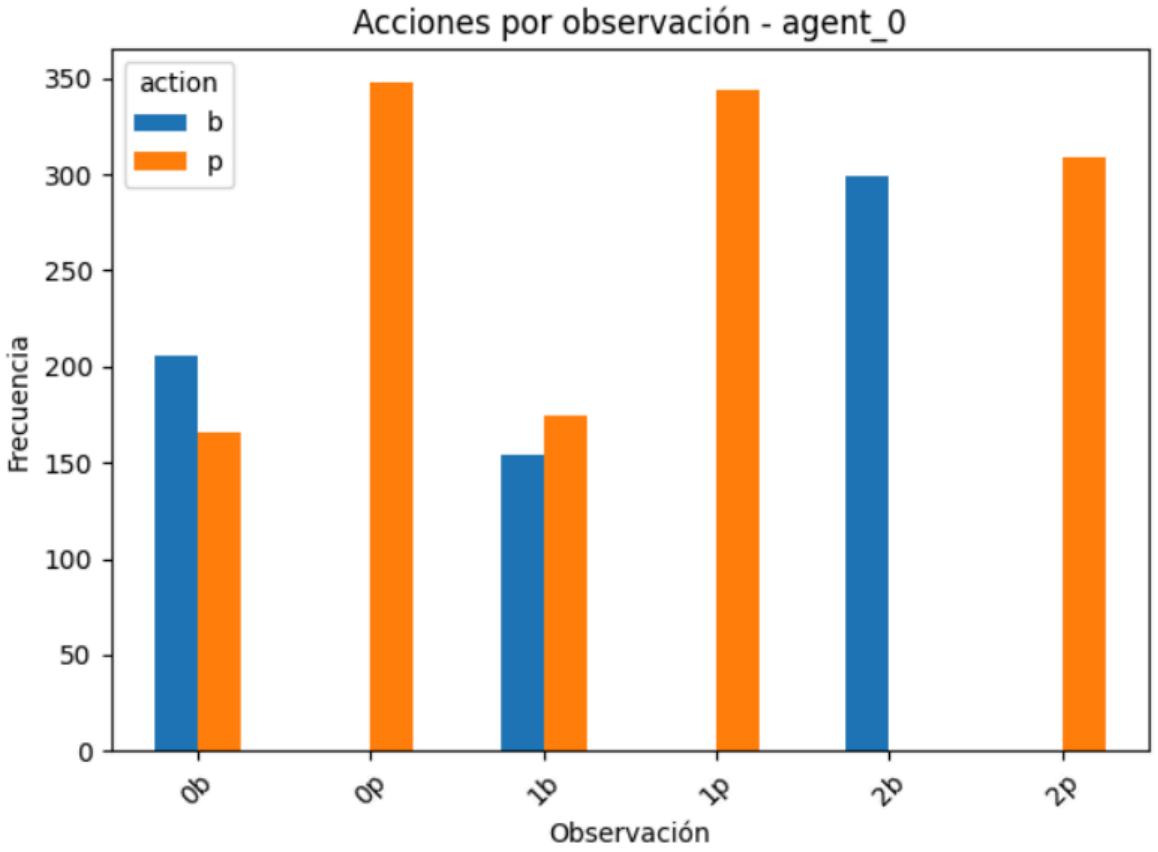
Como intuimos, CFRM es superior a MCTS pero no es una superioridad en cada aspecto. CFRM es mejor en su acumulación de recompensas y MCTS acumula más partidas ganadas. En contextos como un juego de poker esto es preferible al hecho de ganar simplemente, pues no todas las partidas tienen la misma recompensa al ganar la mano. Al parecer MCTS se enfoca en simplemente ganar la partida, mientras que CFRM tiene una mejor percepción sobre el valor de lo que está en juego.



Analizando las acciones por agente vemos que CFRM tiende a comenzar el juego casi que con una misma probabilidad de apostar o pasar indiferente a la carta que le tocó.



	agent	observation	action	count
8	agent_1	0	b	316
9	agent_1	0	p	310
10	agent_1	1	b	362
11	agent_1	1	p	331
12	agent_1	2	b	321
13	agent_1	2	p	360

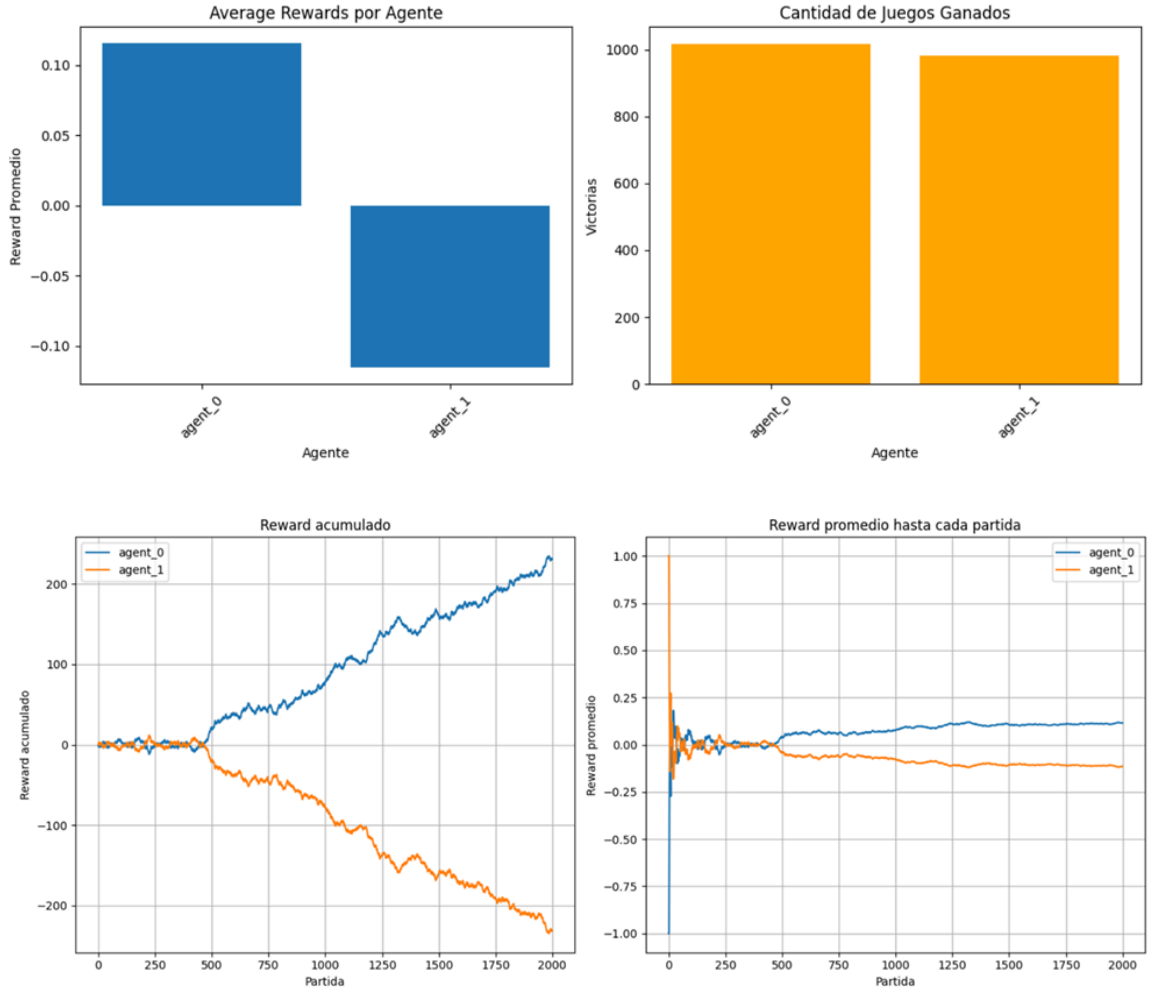


	agent	observation	action	count
0	agent_0	0b	b	206
1	agent_0	0b	p	166
2	agent_0	0p	p	348
3	agent_0	1b	b	154
4	agent_0	1b	p	174
5	agent_0	1p	p	344
6	agent_0	2b	b	299
7	agent_0	2p	p	309

En cambio, para MCTS, las acciones se ven más limitadas y tiende sólo a apostar cuando tiene la mejor carta y CFRM apostó. MCTS tiende a ser mucho más conservador.

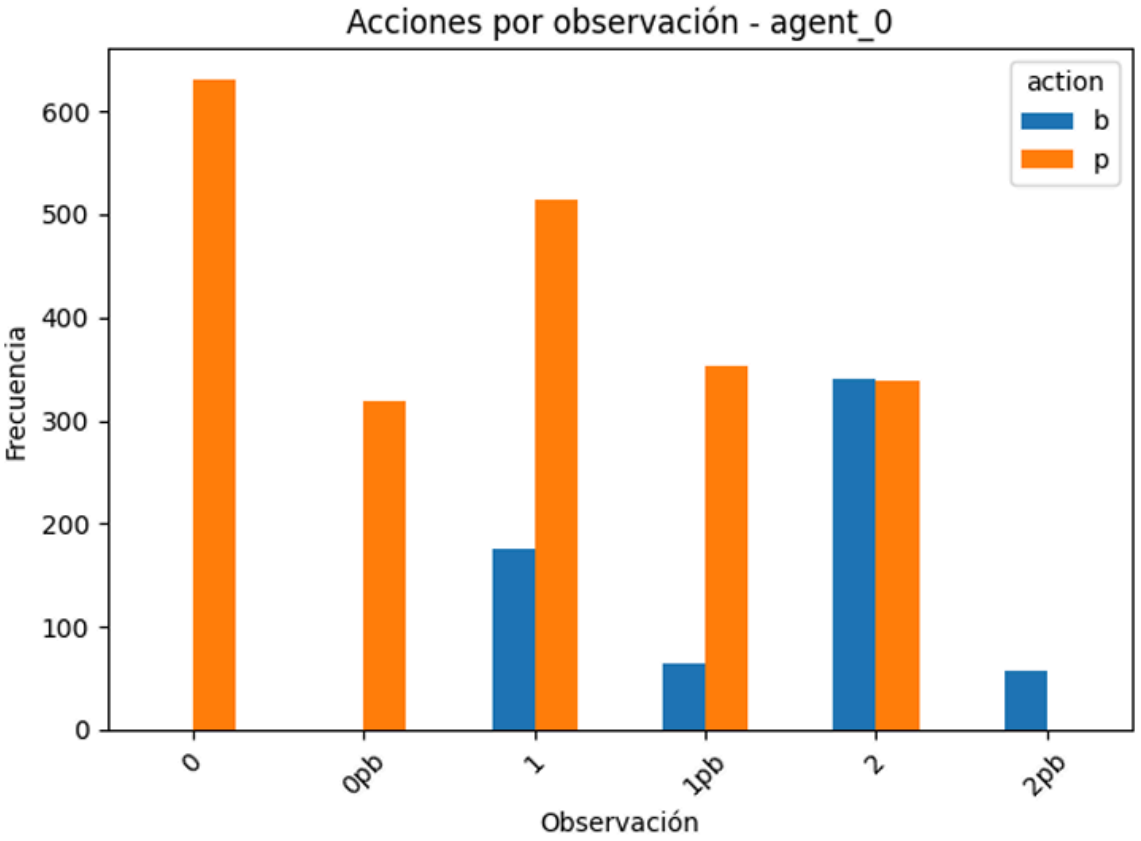
MCTS15 VS CFRM

¿Qué podemos esperar sí aumentamos la cantidad de rollouts a MCTS? Aumentando la cantidad de rollouts en MCTS la situación cambia. Manteniendo la ventaja posicional en en juego para MCTS rollouts = 10 y CFRM con el mismo entrenamiento. Podemos observar lo siguiente:

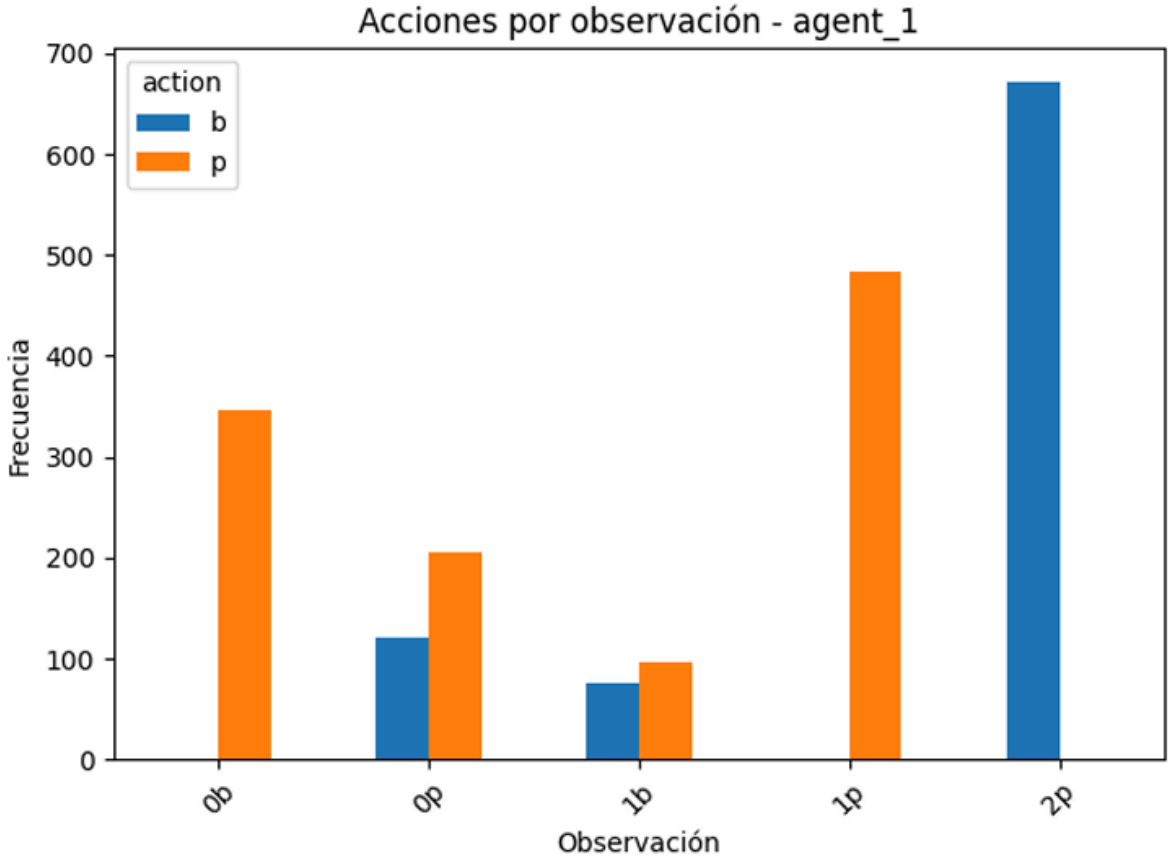


MCTS supera a CFRM luego de 500 iteraciones tanto en juegos como en acumulación de rewards. Podemos intuir que el árbol de decisión de MCTS es lo suficientemente grande como para prever las acciones de CFRM. Sin embargo, cómo se mencionó previamente, a costa de un procesamiento más lento y por ende más costoso. Cabe aclarar que CFRM no sufrió mayores cambios al primer experimento, el entrenamiento se mantuvo igual y se mantuvo en una relativa posición desventajosa en el juego.

En relación a las acciones tomadas por cada agente podemos ver lo siguiente:



	agent	observation	action	count
0	agent_0	0	p	630
1	agent_0	0pb	p	319
2	agent_0	1	b	176
3	agent_0	1	p	515
4	agent_0	1pb	b	65
5	agent_0	1pb	p	353
6	agent_0	2	b	341
7	agent_0	2	p	338
8	agent_0	2pb	b	57



	agent	observation	action	count
9	agent_1	0b	p	346
10	agent_1	0p	b	121
11	agent_1	0p	p	205
12	agent_1	1b	b	75
13	agent_1	1b	p	96
14	agent_1	1p	b	1
15	agent_1	1p	p	484
16	agent_1	2p	b	672

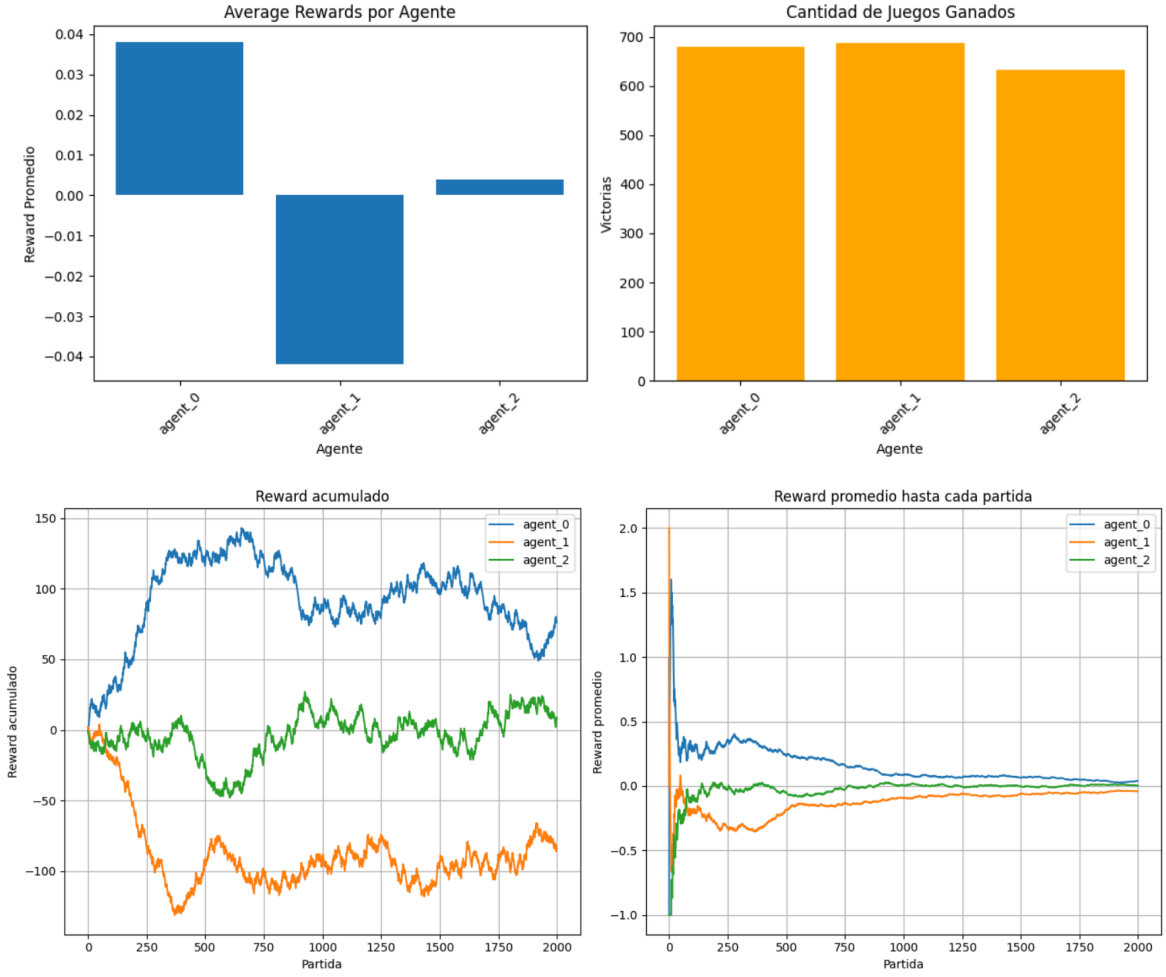
Los juegos son más extensos, hay una dinámica mayor entre los jugadores y no se trunca el juego luego de una única acción. Los agentes tienen observaciones con mayor complejidad y MCTS10 consigue tener una estrategia compleja. Por ejemplo, consigue entender que si tiene una carta 1, pasa y CFRM apuesta (1pb), debe salir del juego, ya que es muy probable que CFRM cuente con un 2, pues CFRM observa: 2p. A pesar de que pierde la partida, toma la mejor decisión a tiempo y logra no tener un reward de -2, y obtiene apenas -1.

Póker de Kuhn para tres jugadores

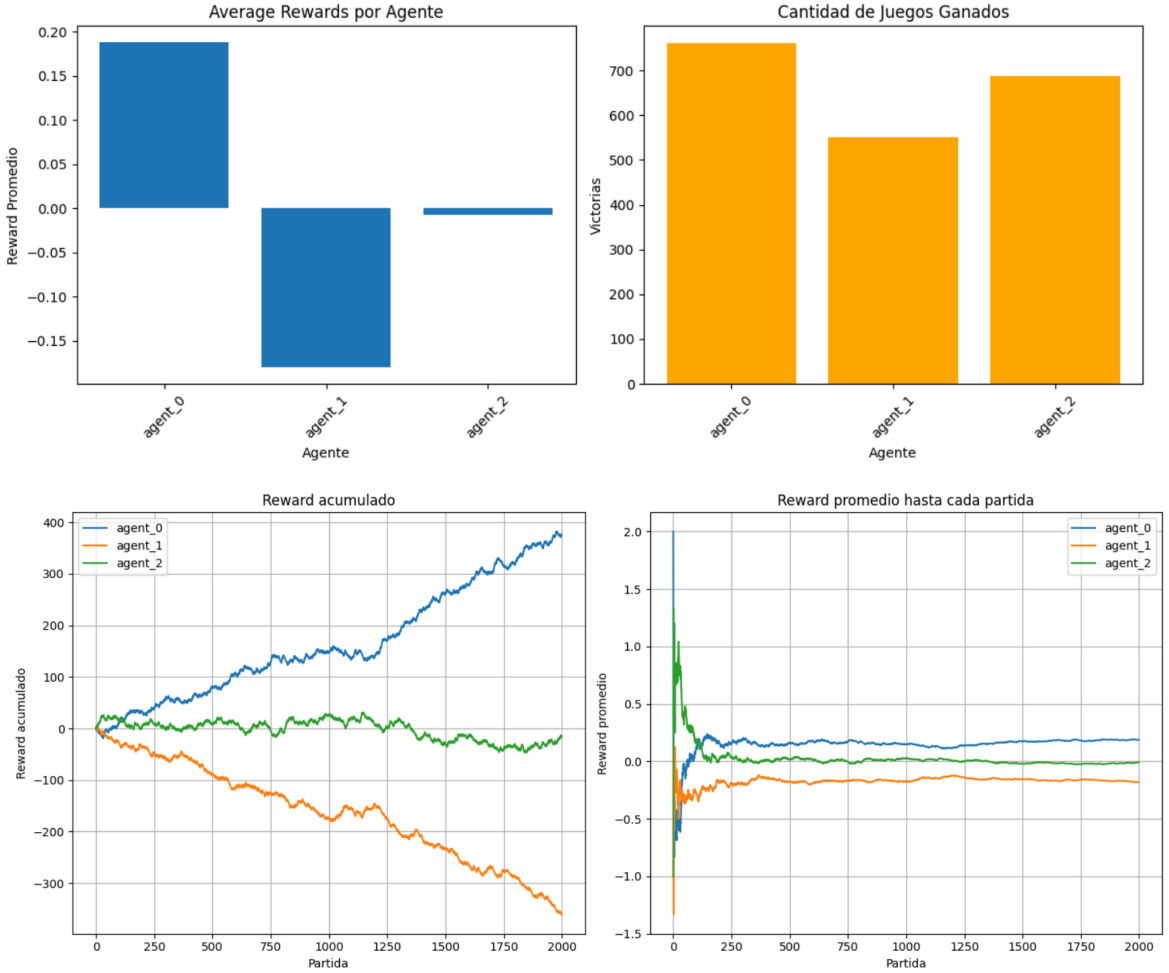
¿Qué podemos esperar si aumentamos el número de jugadores en Póker de Kuhn? ¿Lo descubierto para el contexto anterior se mantiene, o evoluciona?

CFRM VS CFRM VS CFRM

Como observamos en el Póker de Kuhn de dos jugadores, la posición es fundamental pues puede significar la ventaja de un agente sobre otro. Por eso, es necesario realizar algún experimento para encontrar cuál es la posición que se beneficia en un juego de tres. Para ello, realizamos los siguientes experimentos: uno con 3 agentes CFRM y otro con 3 agentes MCTS con rollouts = 2.



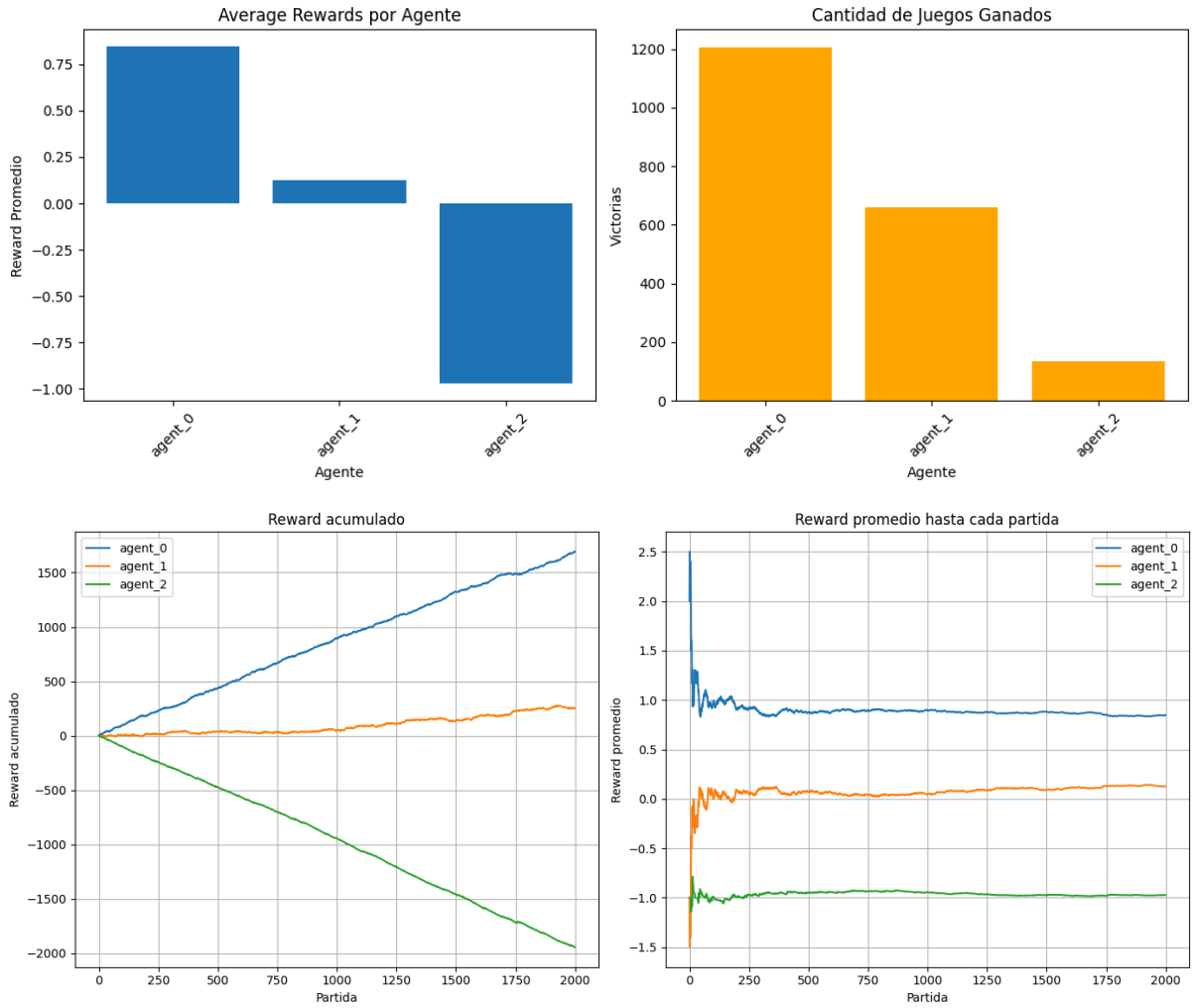
MCTS VS MCTS VS MCTS



Podemos observar lo siguiente, para MCTS la posición 1 es preferible a las posiciones 2 y 3. Mientras que para CFRM, la mejor posición es la número 1 y 3.

CFRM VS MCTS VS MCTS

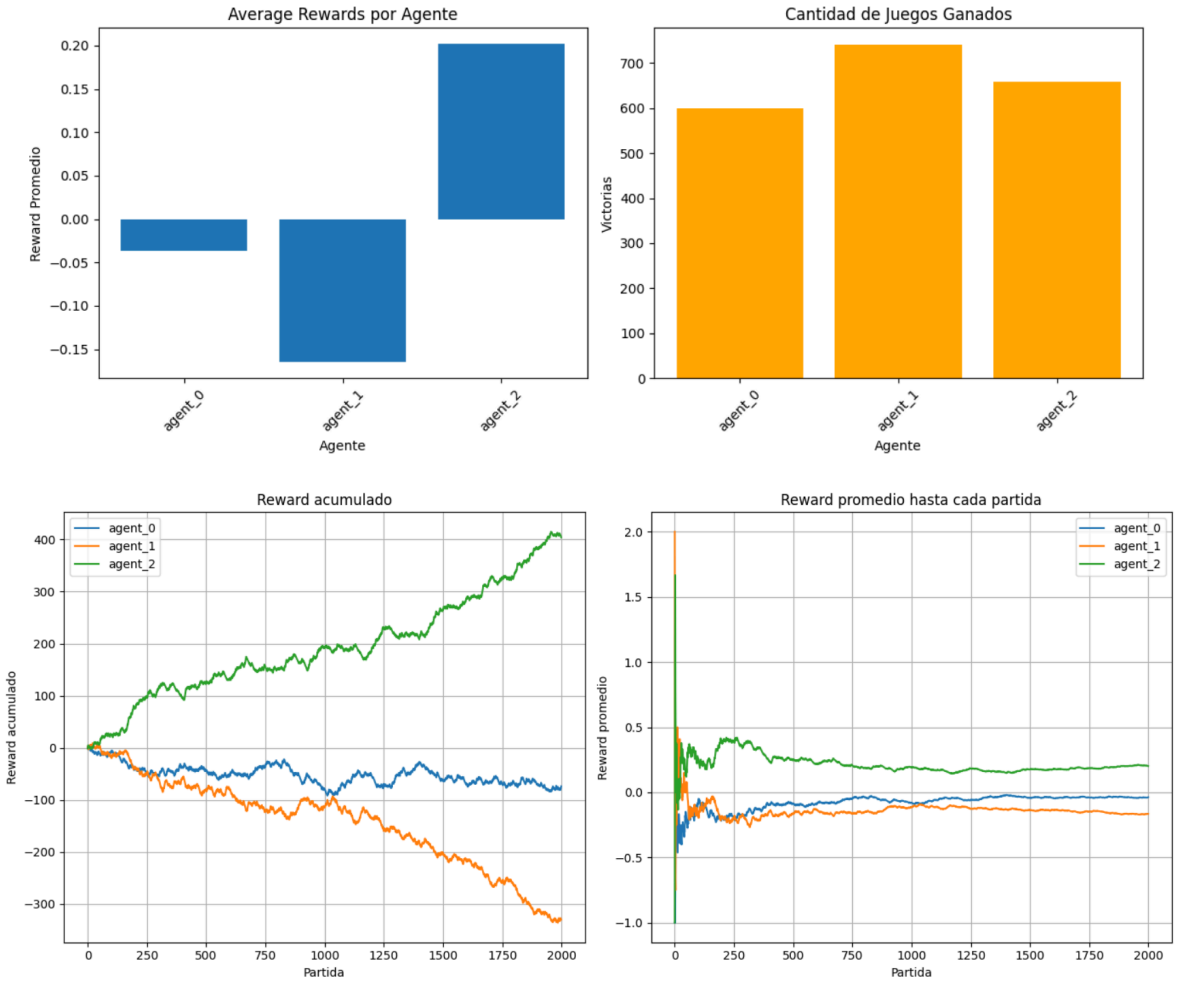
¿Es CFRM un mejor algoritmo ante dos MCTS? ¿Se mantiene su superioridad ante dos oponentes? El siguiente experimento propone a agente_0 como CFRM mientras que para las otras dos posiciones tenemos a MCTS con rollout = 2. Sucedió lo siguiente:



Como observamos, desde el primer momento CFRM es superior a sus competidores desde el primer momento. A pesar de que todos los agentes cuentan con la mejor posición posible, ninguno de los agentes MCTS2 no son suficientemente extensos como para prever las posibles acciones de CFRM.

CFRM VS MCTS5 VS MCTS5

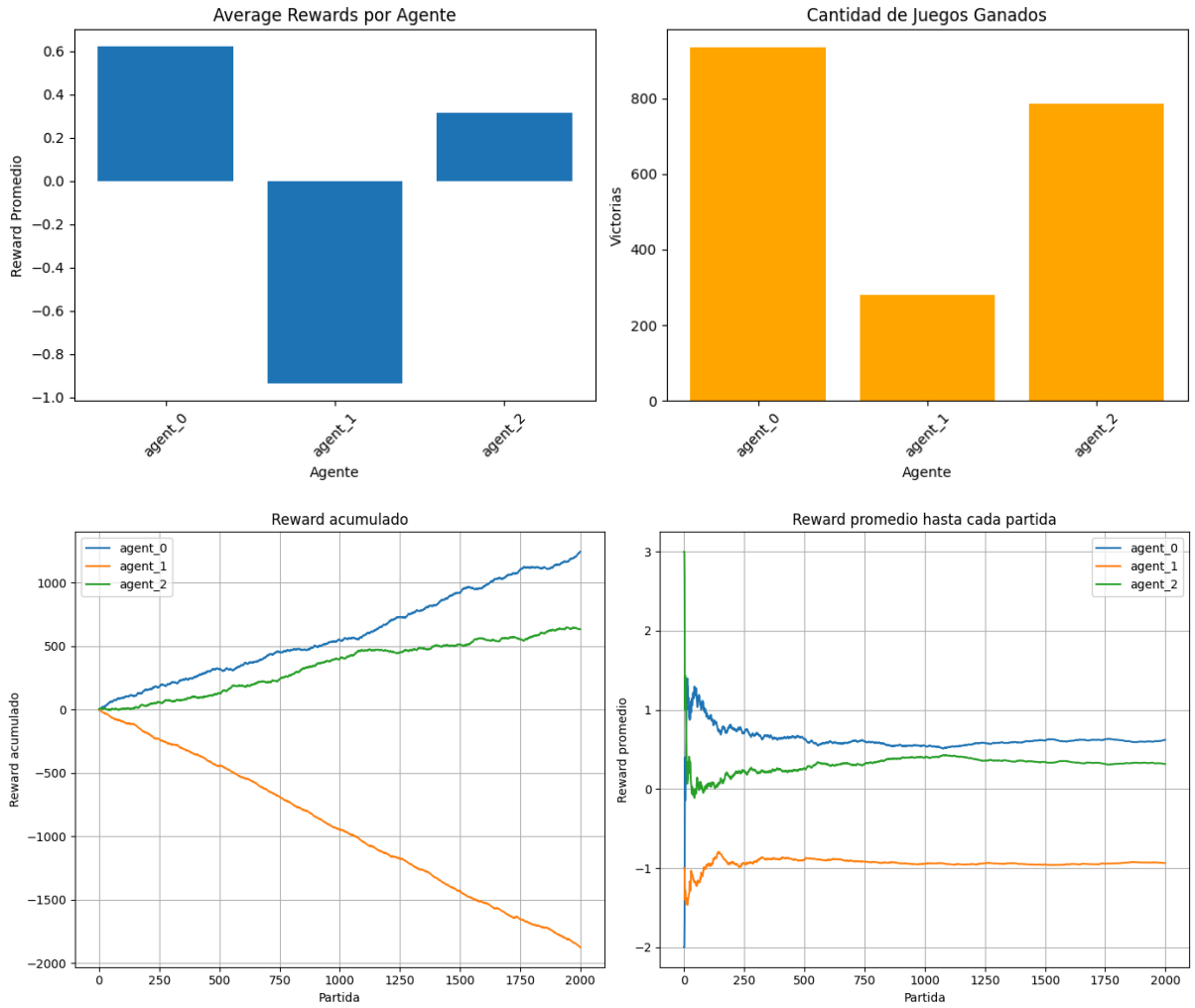
¿Qué sucede si hacemos lo mismo pero ante dos MCTS rollaout = 5? A diferencia del entorno para dos jugadores, CFRM no consigue sobreponerse ante dos agentes MCTS5. Inclusive, como observamos a continuación, es ampliamente vencido ante estos agentes:



El escenario anterior dónde CFRM era superior a un único MCTS5 no se repitió. CFRM aún cuenta con los 10000 episodios de entrenamiento previo y no hacemos cambios sobre el agente al probarlo ante diferentes versiones de MCTS.

MCTS5 VS Random VS CFRM80K

Al probar con un entrenamiento de 80.000 iteraciones para CFRM vemos que el resultado mejora aunque continúa teniendo dificultades para ser el mejor agente del juego. MCTS5 continúa ganando frente a CFRM en un juego de tres competidores. No obstante podemos intuir que aumentar el número de iteraciones de entrenamiento para CFRM hará que eventualmente sea mejor para este entorno propuesto.



Una de las posibles causas de este resultado es el hecho de que CFRM es entrenado contra otro agente igual. No se realiza un entrenamiento con un agente MCTS y pues, una vez que el agente actúa dentro de un entorno online, la política aprendida puede no ser la más óptima y, de este modo, tenga mayores dificultades sí no se realiza ninguna actualización de política durante la fase online. Aumentar el número de iteraciones de entrenamiento podría no ser suficiente y debería de realizarse otro tipo de aproximación para vencer en este contexto. No obstante, cabe resaltar que los tiempos de inferencia son ampliamente superiores a un MCTS y es difícil que en entornos reales se utilicen algoritmos tan lentos para tomar una decisión. únicamente contextos muy particulares admitirán esta espera.

Conclusión

El Póker de Kuhn es un entorno ideal para evaluar distintos algoritmos en juegos de suma cero con información imperfecta. Tanto MCTS como CFRM son estrategias válidas para abordar este tipo de juego, aunque cada una presenta ventajas y desventajas particulares.

Como se ha observado, MCTS no es necesariamente un mal algoritmo para competir en el Póker de Kuhn. Sin embargo, su desempeño está fuertemente condicionado por su configuración inicial. Hemos hecho pruebas exclusivamente sobre los rollouts pero existen otras configuraciones que pueden mejorarlo aún más, por ejemplo mayor número de simulaciones. Una configuración más extensa incrementa la capacidad del agente para anticipar escenarios posibles, pero también implica un mayor costo computacional. Esto puede traducirse en tiempos de ejecución muy elevados: una partida de 2000 iteraciones puede demorar más de una hora. En este sentido, MCTS puede considerarse una solución de fuerza bruta cuando no se aplica ninguna técnica de optimización como por ejemplo: una poda.

Por otro lado, CFRM se presenta como una alternativa más eficiente en términos computacionales durante la ejecución. Al basarse en el arrepentimiento contrafactual, no necesita construir árboles para explorar escenarios posibles. Esto le permite actuar de forma directa y rápida, siempre que haya sido previamente entrenado. Su principal desventaja radica en este punto: requiere una fase de entrenamiento antes de ser competitivo, lo cual no siempre es viable dependiendo del entorno o las restricciones del problema. Otra desventaja estaría relacionado al entrenamiento, pues los entrenamientos se hacen pensando en un entorno y oponentes determinados. Al llevarlo a una fase productiva (u online), podría ser diferente a la fase de entrenamiento.

En suma, la elección de un agente dependerá de los recursos disponibles y del objetivo. Si se requiere adaptar un agente sin un entrenamiento previo, MCTS puede ser la solución. En cambio, si hay espacio para entrenar el agente, CFRM ofrece una solución más eficiente y estratégica.

Bibliografía

- **OpenAI. (2025).** ***ChatGPT*** (versión GPT-4o) [Modelo de lenguaje]. https://chat.openai.com/chat
- **Albrecht, S. V., Christianos, F., & Schäfer, L. (2024)**. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press. https://www.marl-book.com/
- **S. Russell, P. Norvig.** Artificial Intelligence. A modern approach. 4th ed. Prentice Hall.
- **John Billingham.** Simplified three player Kuhn poker. https://arxiv.org/pdf/1704.08124
- **D. Szafron, R. Gibson, N. Sturtevant.** A Parameterized Family of Equilibrium Profiles for Three-Player Kuhn Poker. https://www.ifaamas.org/Proceedings/aamas2013/docs/p247.pdf
- ***multiagentes-alt-games*** **[Repositorio de código]**. Disponible en: https://github.com/leandrouruguay1994/multiagentes-alt-games