

Sistemas Multiagente

Obligatorio 1

Universidad ORT, Uruguay

Integrantes:

Martin Rizzo - 343631
Leandro Cardoso - 166267

Índice

Índice	2
Introducción:	3
Primera parte: N-Form Games	3
Matching Pennies	4
FP vs RA en MP	4
RM vs RA en MP	6
FP vs RM en MP	7
Rock, Paper and Scissors	9
FP vs RA en RPS	9
RM vs RA en RPS	11
FP vs RM en RPS	12
Blotto	14
Blotto(5, 2)	14
Blotto(7, 3)	14
FP vs RA en Blotto(7, 3)	14
RM vs RA en Blotto(7, 3)	16
FP vs RM en Blotto(7, 3)	17
Blotto(12, 4)	19
FP vs RA en Blotto(12, 4)	19
RM vs RA en Blotto(12, 4)	20
FP vs RM en Blotto(12, 4)	22
Segunda Parte: Stochastic Games	24
Foraging	24
Independent Q-Learning	25
Juegos competitivos de IQL	26
Foraging 5x5 2v2: IQL vs IQL	26
Foraging 5x5 2v2: IQL vs IQL vs IQL	26
Foraging 8x8 2v2: IQL vs IQL	27
Juegos cooperativos de IQL	28
Foraging 5x5 2v2 Coop: IQL vs IQL vs IQL	28
Joint Action Learning - Agent Modeling	29
Juegos competitivos de JAL-AM	29
Foraging 5x5 2v2: IQL vs JAL-AM	29
Foraging 5x5 2v2: JAL-AM vs JAL-AM	30
Foraging 5x5 3v3: IQL vs JAL-AM vs JAL-AM	31
Foraging 5x5 3v3: IQL vs IQL vs JAL-AM	31
Juegos cooperativos de JAL-AM	32
Foraging 5x5 3v3 Coop: IQL vs JAL-AM vs JAL-AM	33
Foraging 5x5 3v3 Coop: IQL vs IQL vs JAL-AM	34
Conclusiones	35
Bibliografía	36

Introducción:

En el siguiente informe se muestran los experimentos realizados en los ambientes implementados para N-Form Games como Matching Pennies, Rock, Paper, Scissors y Blotto, así como para Foraging correspondiente a juegos estocásticos. Para el primer caso se prueban los agentes implementados regret matching, fictitious play y también un random agent, que toma acciones aleatoriamente. Para el caso de Foraging se prueban los agentes Independent Q Learning y Joint Action Learning con Agent Modeling en distintas configuraciones de este juego.

Para cada juego se corren agentes distintos para comparar y observar los resultados. En el caso de Foraging también se corren combinaciones de agentes tanto en su modalidad competitiva como en su modalidad cooperativa.

Primera parte: N-Form Games

Los N-form games están compuestos por tres ambientes. Estos son: Matching Pennies (MP); Rock , Paper and Scissors(RPS) y Blotto.

En los ambientes mencionados, colocaremos agentes a competir entre ellos con el objetivo de que obtengan el mejor resultado posible. Para estos ambientes los agentes válidos que utilizaremos son: Fictitious Play (FP), Regret Matching (RM) y Random Agent (RA).

Dados estos agentes, las combinaciones válidas posibles para cada uno de estos ambientes son: FP vs RA, FP vs RM y RM vs RA.

Para cada uno de nuestros análisis utilizaremos 100000 episodios de entrenamiento, y luego procederemos a generar diez episodios de prueba con las políticas aprendidas durante el entrenamiento para cada agente. De esta forma obtenemos el resultado de cada una de las rondas en test y gráficos de:

- Distribución de acciones tomadas por cada agente
- Recompensa acumulada en test por cada agente
- Política aprendida

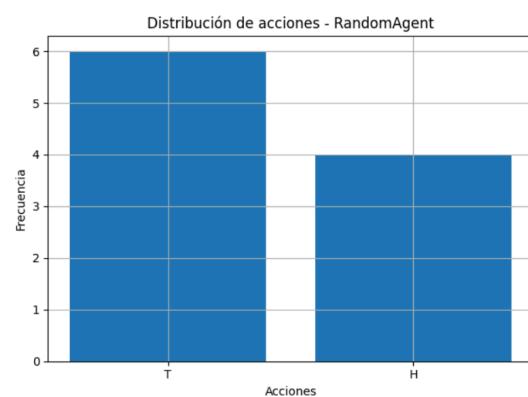
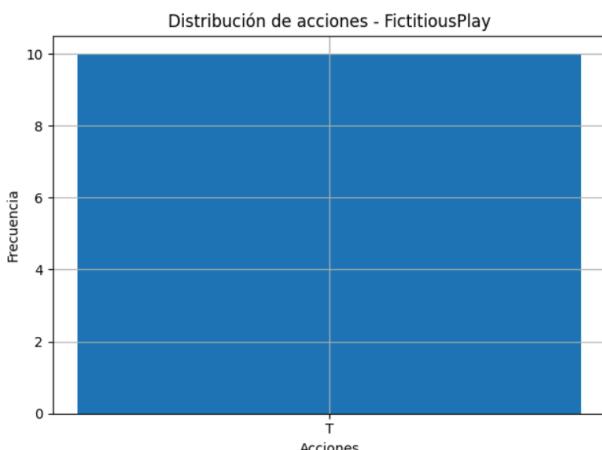
A partir de este punto, desarrollaremos nuestro análisis sobre cada entorno y los enfrentamientos descritos, con el objetivo de extraer conclusiones relevantes.

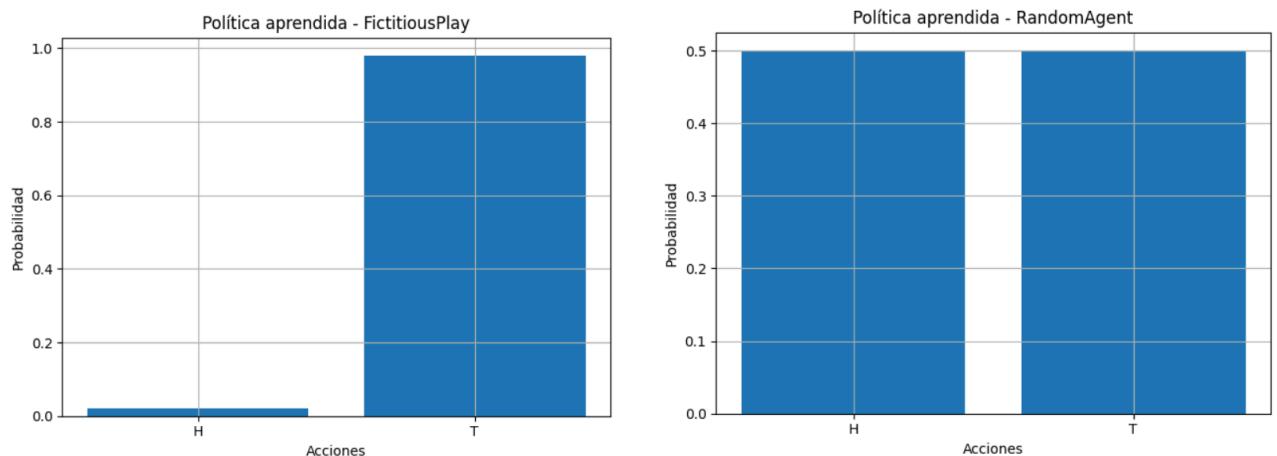
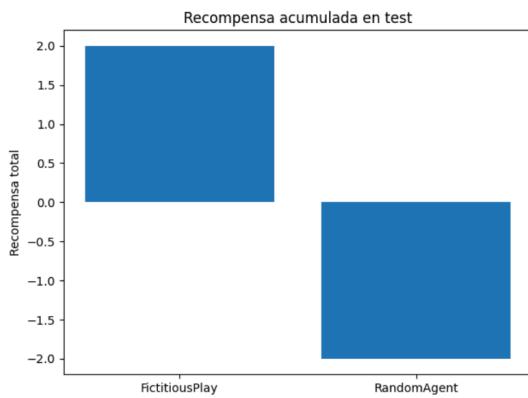
Matching Pennies

Es un ambiente dónde dos agentes juegan seleccionando dos acciones posibles: head (H) o tails (T). El agente ganador es aquel que acierta la opción (H o T) determinada por el entorno.

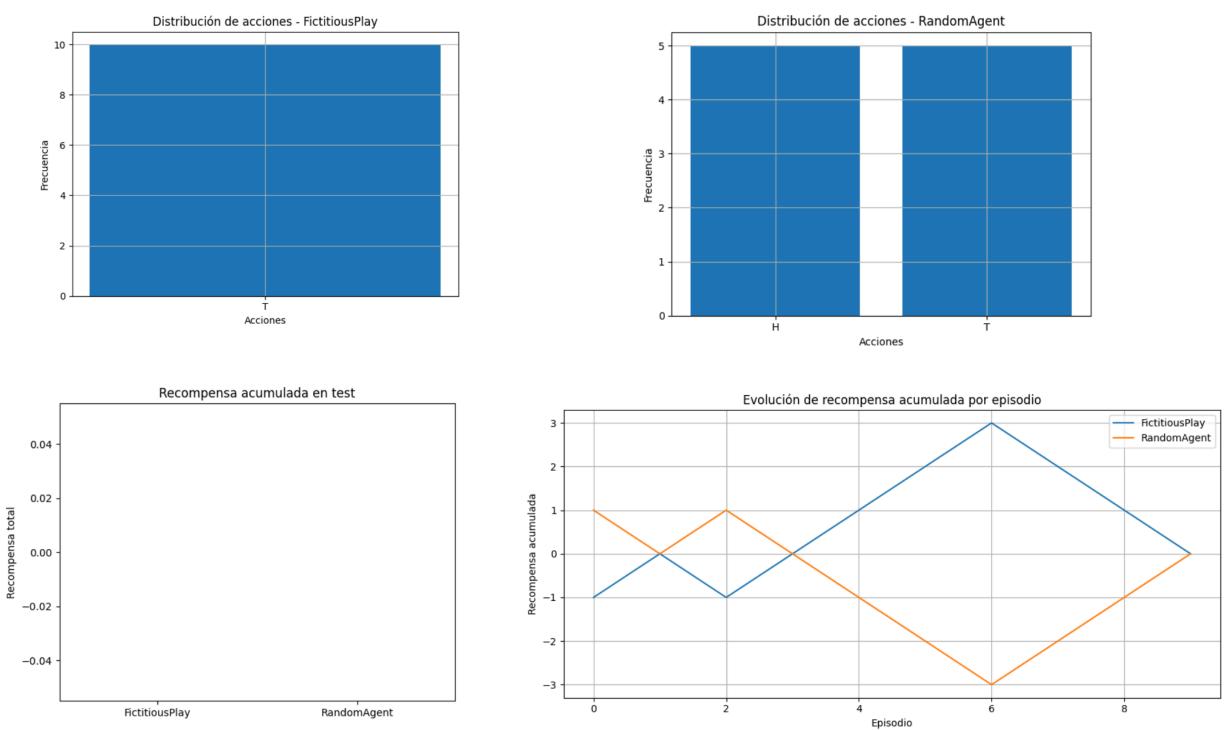
FP vs RA en MP

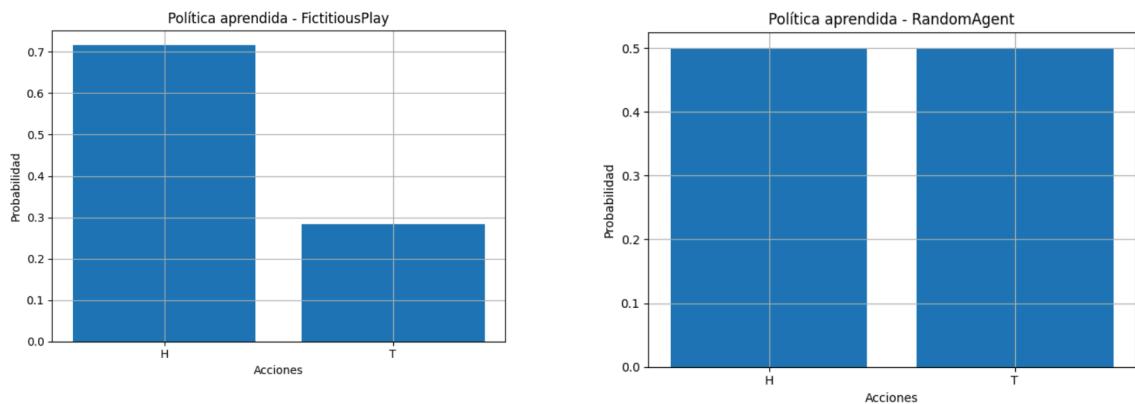
Cuando RA compite contra FP, éste primero consigue superarlo si la toma de decisiones se inclina hacia alguna de las dos posibles. FP selecciona sistemáticamente la mejor respuesta aprendida durante la fase de entrenamiento, dejando un margen pequeño de opciones. En ambientes dónde existen pocas opciones de acción esta puede no ser el mejor escenario, en este caso FP consigue vencer a RA al conseguir un reward acumulado mayor. Si RA elige más veces una mejor respuesta, tiene probabilidad de superarlo.





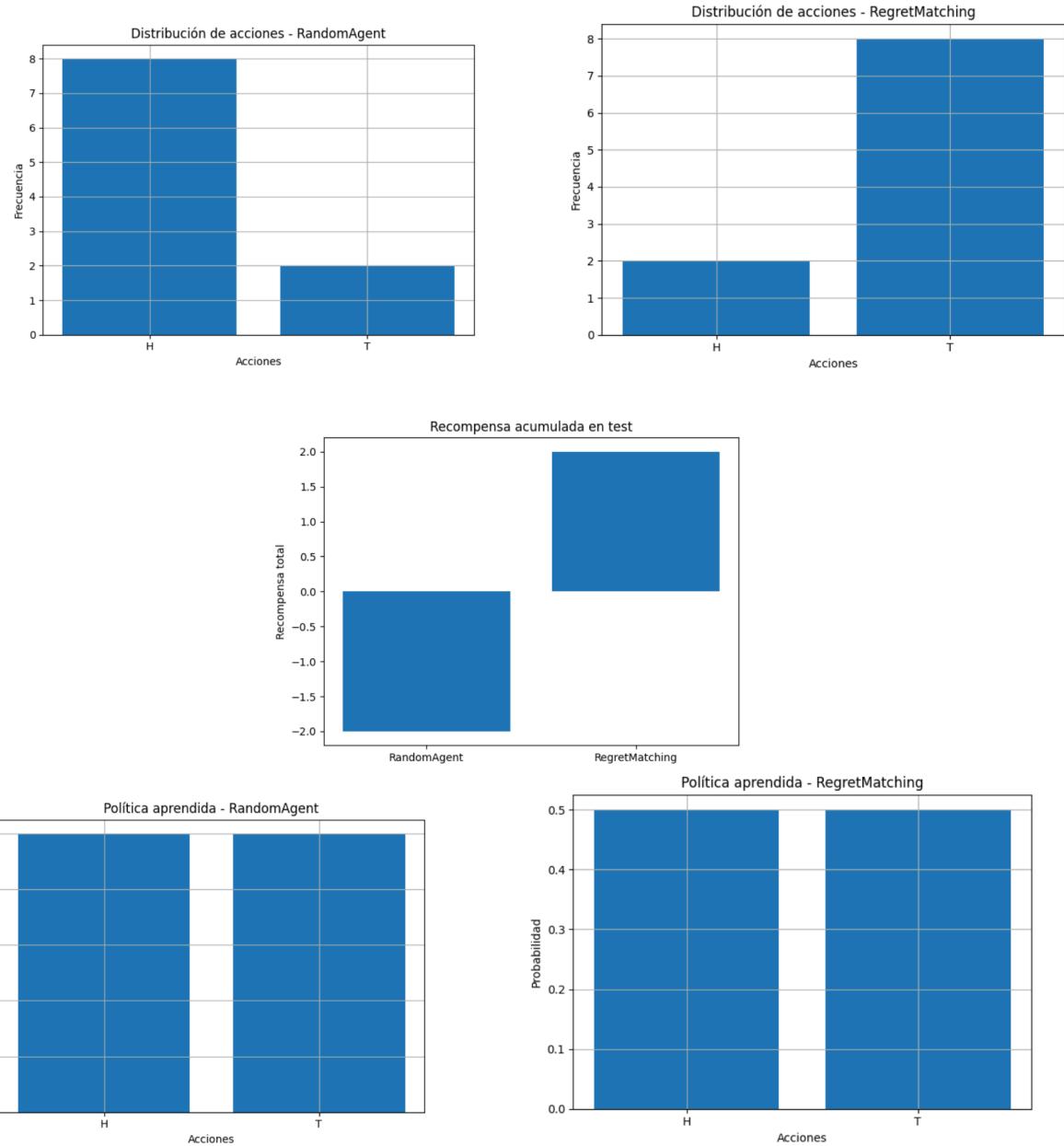
En una segunda ejecución realizada, vemos que FP supera a RA. Esto sucede una vez que RA distribuye sus acciones igualitariamente. Es ahí donde se consigue un equilibrio en el reward acumulado entre ambos agentes.



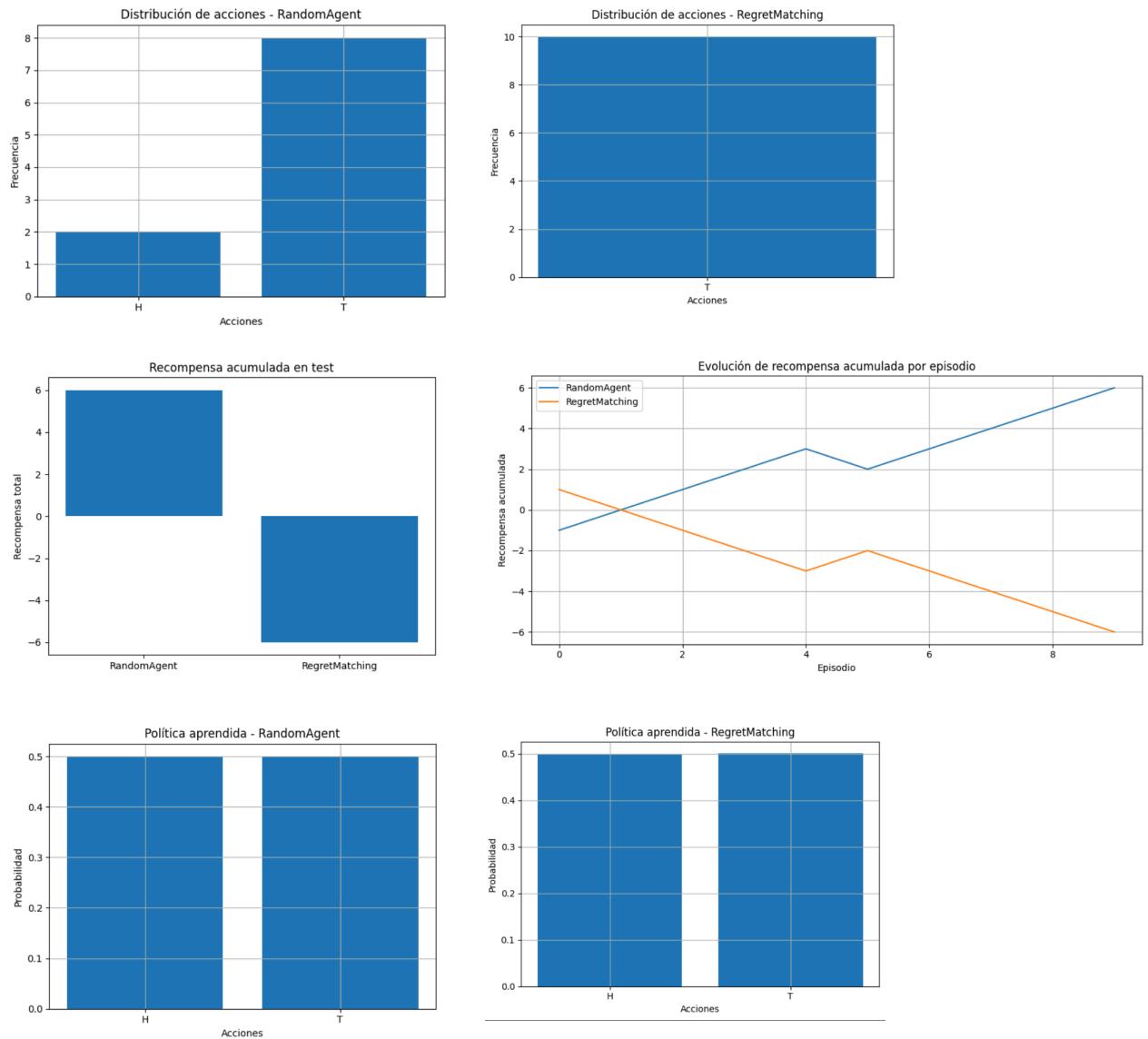


RM vs RA en MP

En relación del cruce entre RM y RA, en nuestros tests, RM termina aprendiendo una política igual a RA. No obstante, durante los juegos realizados durante test es RM quien termina venciendo a RA a partir de jugar acciones espejo.

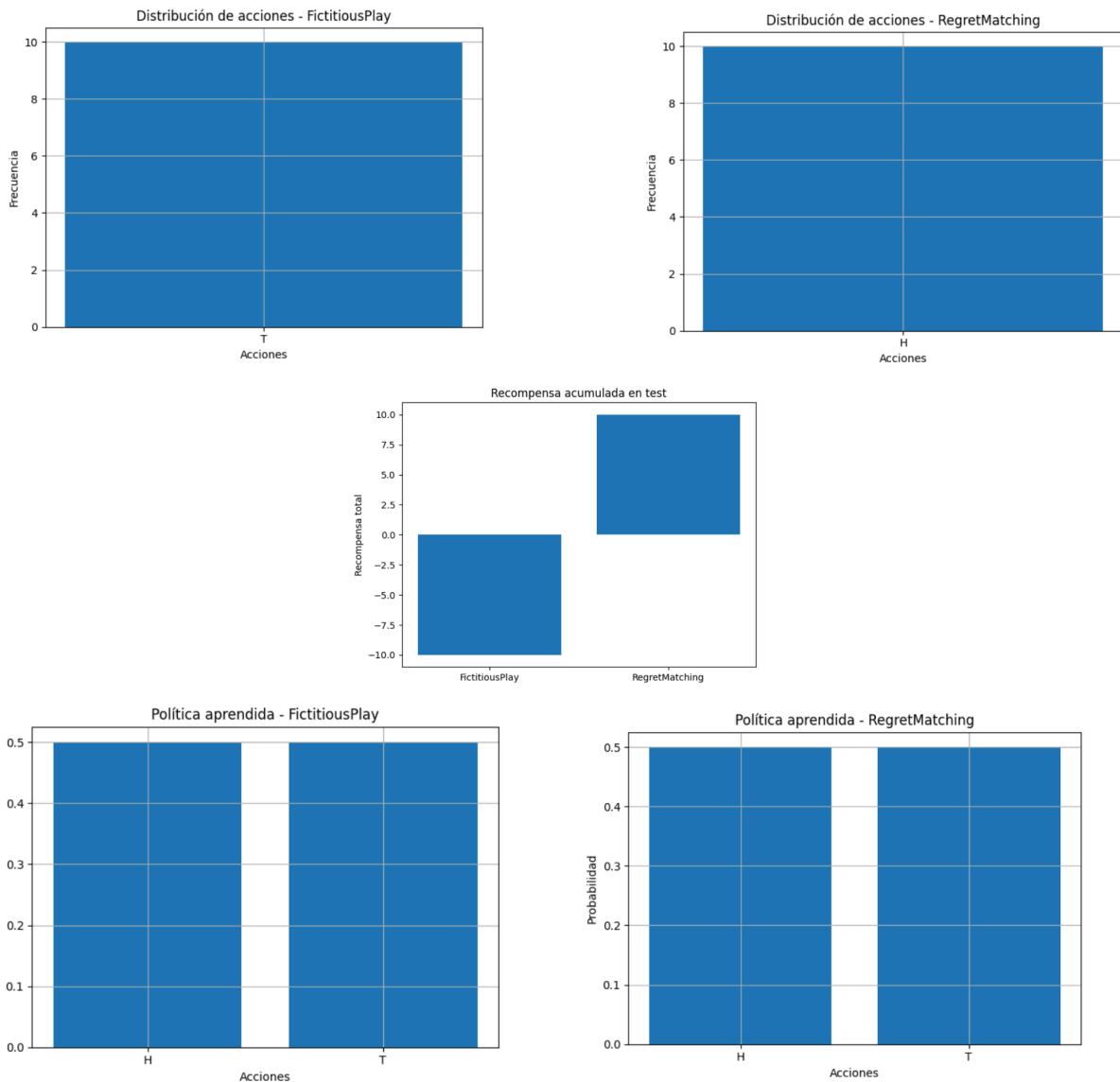


Al realizar un segundo test entre RM y RA. Vemos que RA continua superando a RM. Obteniendo los mismos resultados que para el primer test.

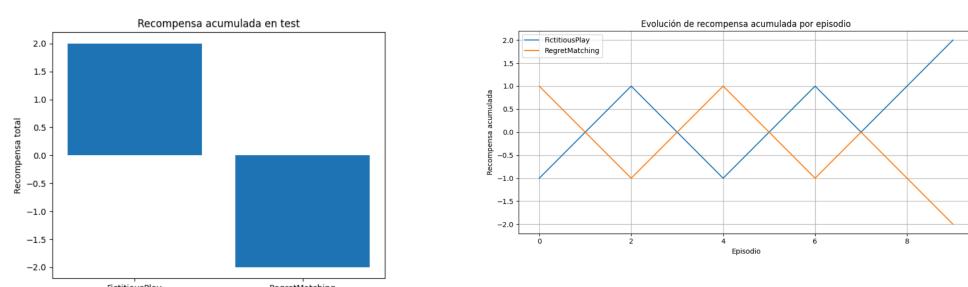
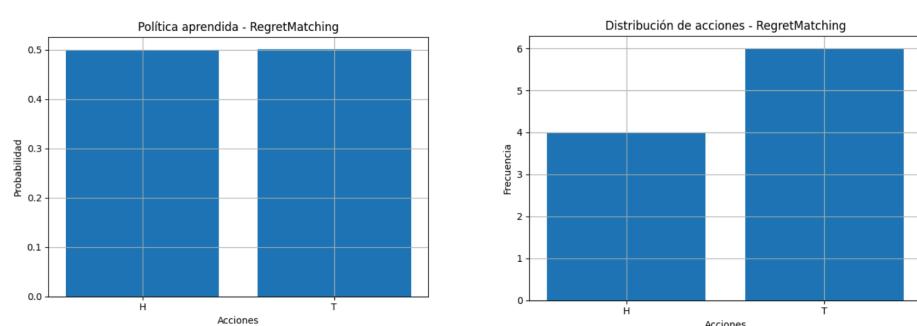


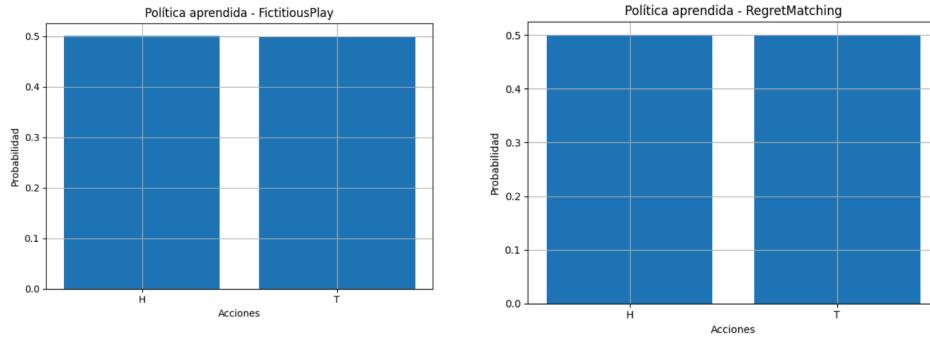
FP vs RM en MP

Para el caso de cuando FP juega contra RM en test, ambos toman la misma decisión de jugar una única acción opuesta a la de su contrincante. En nuestra prueba es RM quien sale victorioso. Vemos que ambos agentes llegan a un set de políticas de acciones, FP no consigue definir una política que sea la mejor para jugar contra RM, pues la naturaleza de RM es más explicativa a causa de su factor de *regret*. Este factor permite una mayor flexibilidad y consigue adaptarse para ganar frente a FP



Al realizar un segundo test, vemos que obtenemos las mismas distribuciones de acciones aprendidas para cada agente. Tanto FP como RM tienen una posibilidad del 50% de elegir una u otra acción. No obstante, FP continua seleccionando una de las acciones de forma sistemática mientras que RM varía .



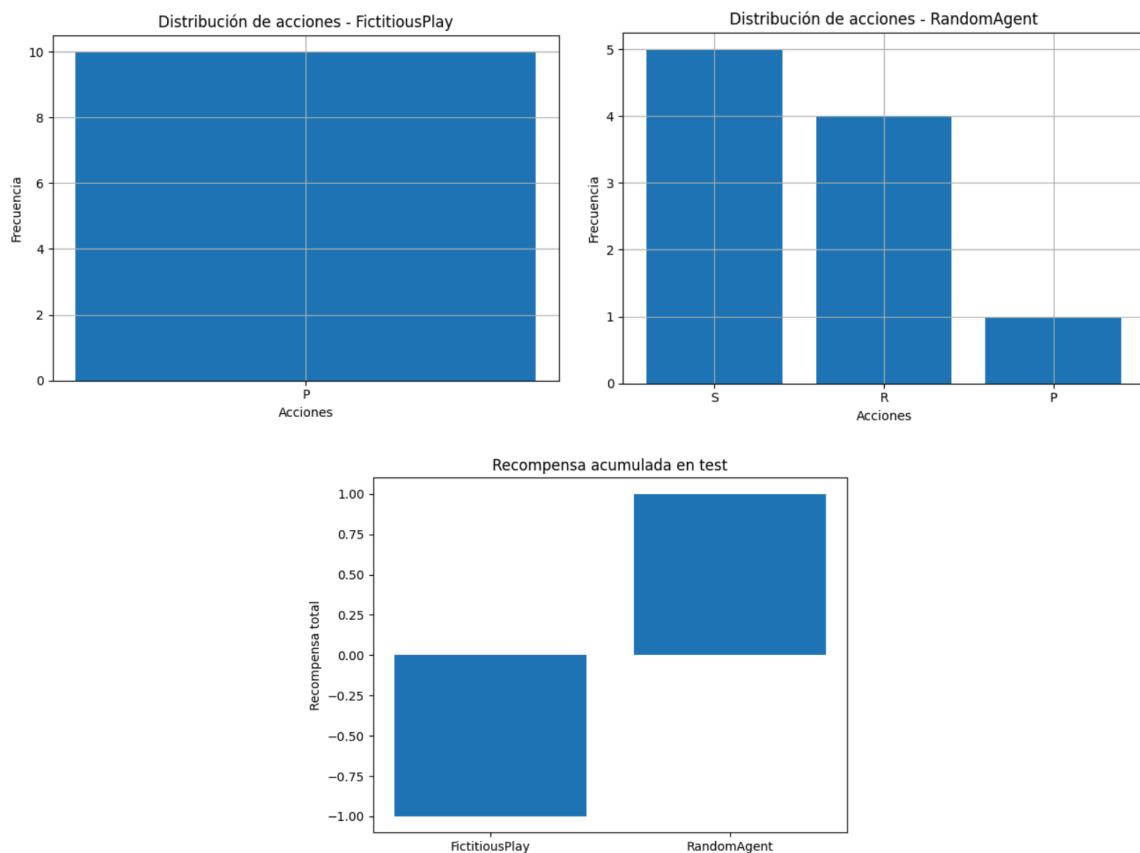


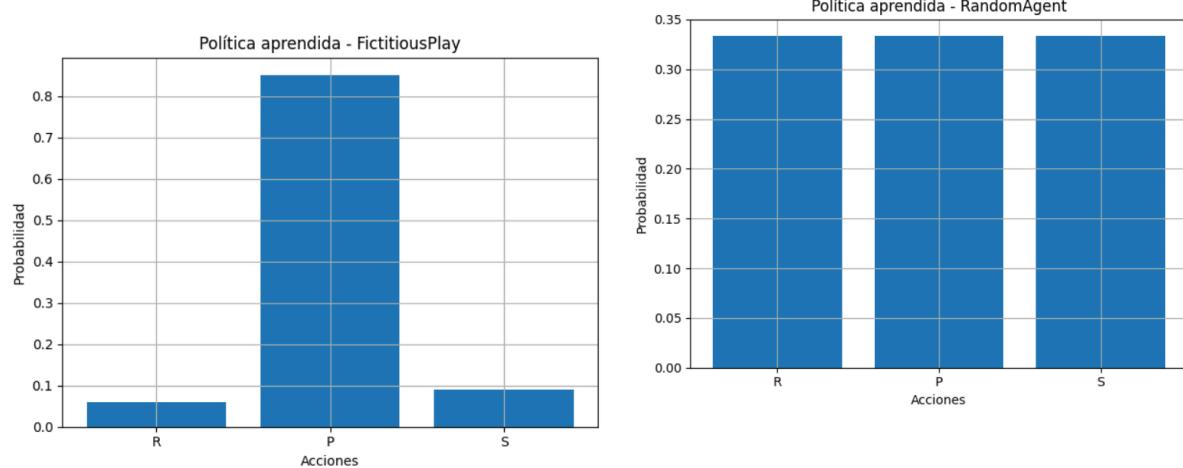
Rock, Paper and Scissors

El ambiente RPS es la adaptación del popular juego infantil piedra, papel o tijeras. Ahora en vez de tener dos acciones posibles, pasamos a tener tres acciones posibles: paper (P), rock (R) y scissors (S). Este ambiente es un más amplio que el previo al contar con tres acciones posibles.

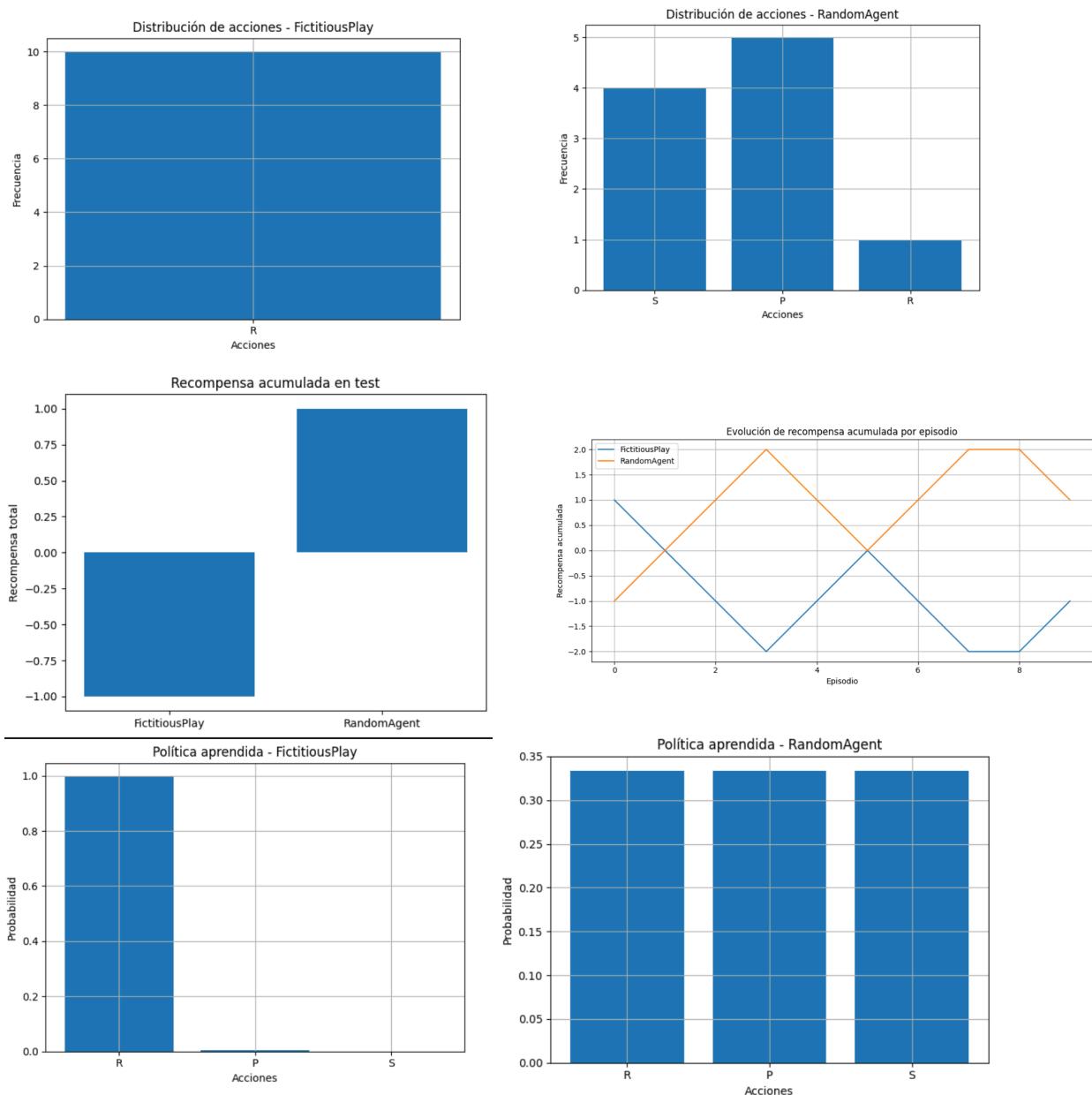
FP vs RA en RPS

En RPS, FP continua actuando de la misma forma. Escoge una única acción sistemáticamente durante la fase de test. No es que su set de políticas sea única, sino que existe una acción exponencialmente más probable de ser tomada que las otras. Al enfrentarse a RA, es RA quien lo supera al contar con una posibilidad mayor de seleccionar una acción diferente.



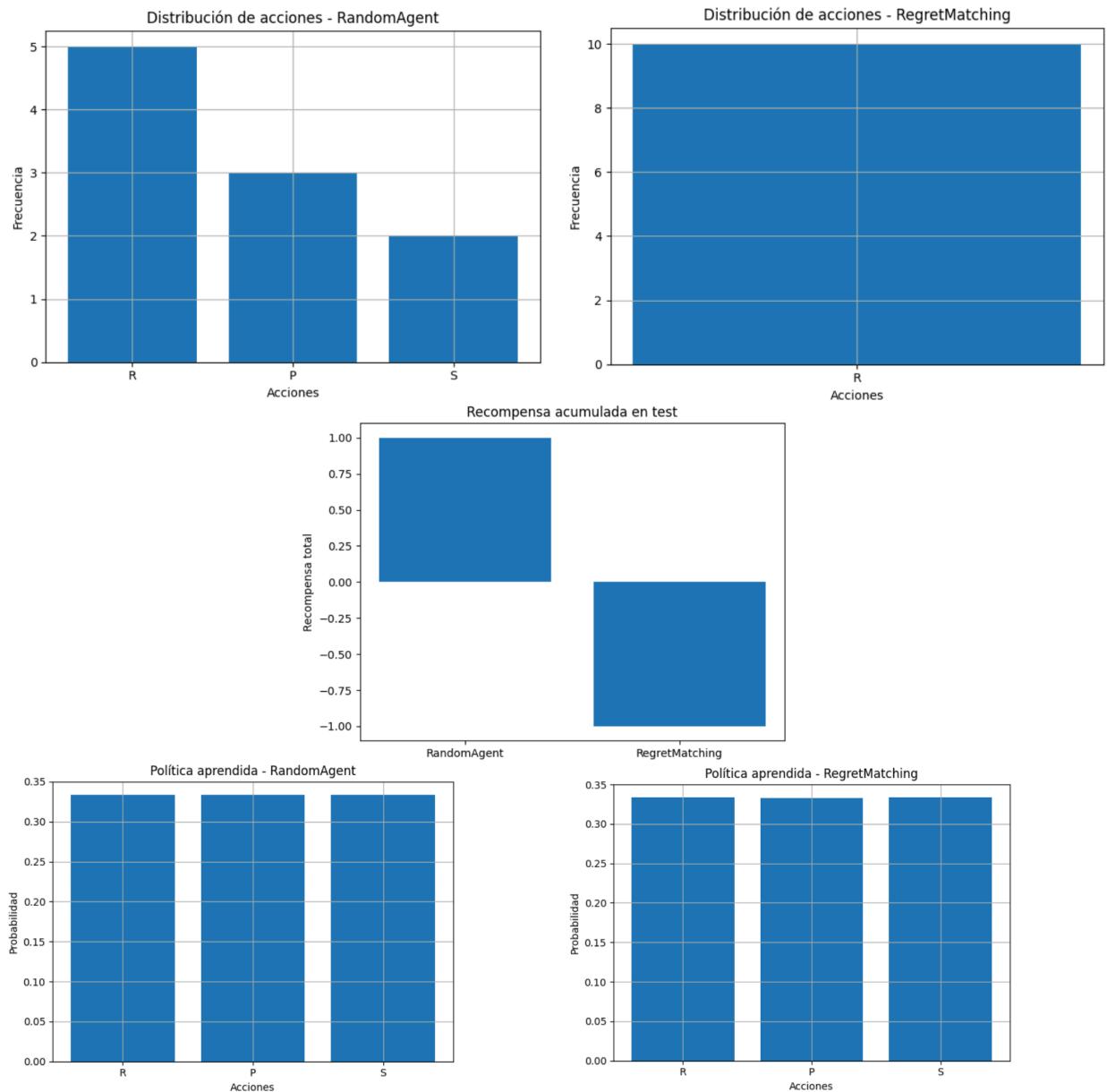


En una segunda prueba vemos un comportamiento similar. FP se dedica a aprender una política con una acción dominante, para luego enfocarse a ejecutar una única acción durante la prueba. Esto genera que RA salga victorioso al contar con más opciones.

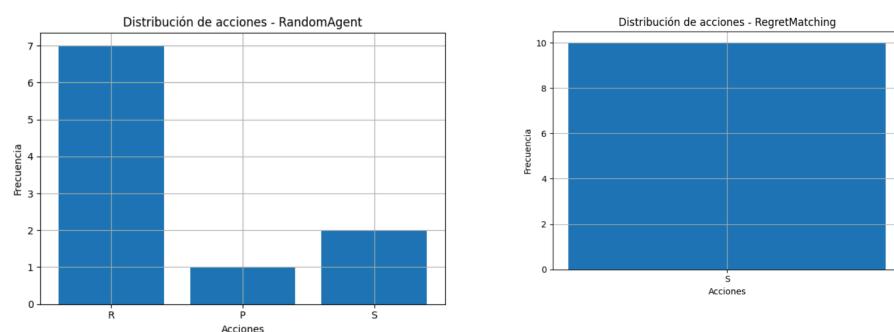


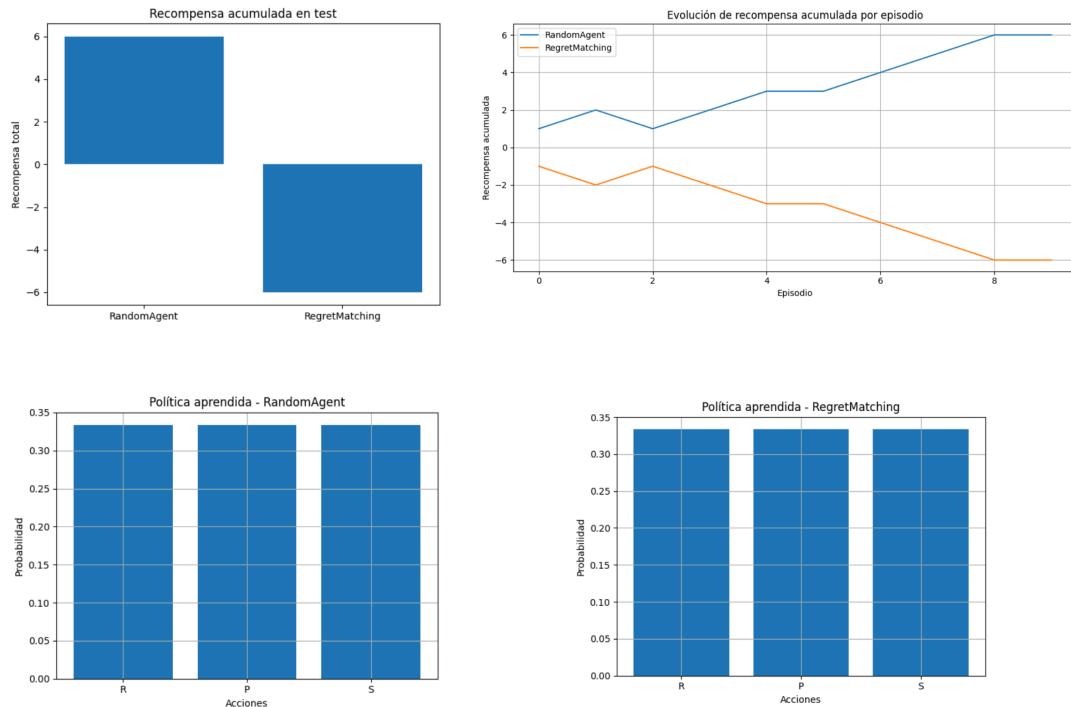
RM vs RA en RPS

Para el juego entre RM y RA, vemos que RM aprende la misma política que RA. Todas las acciones tienen la misma probabilidad de ser escogidas durante test. En nuestros tests, RM eligió siempre la misma acción y terminó con un reward acumulado menor a RA. Podríamos observar que es una situación similar a la anterior con FP, donde la rigidez de acción causó que el reward acumulado final sea menor.



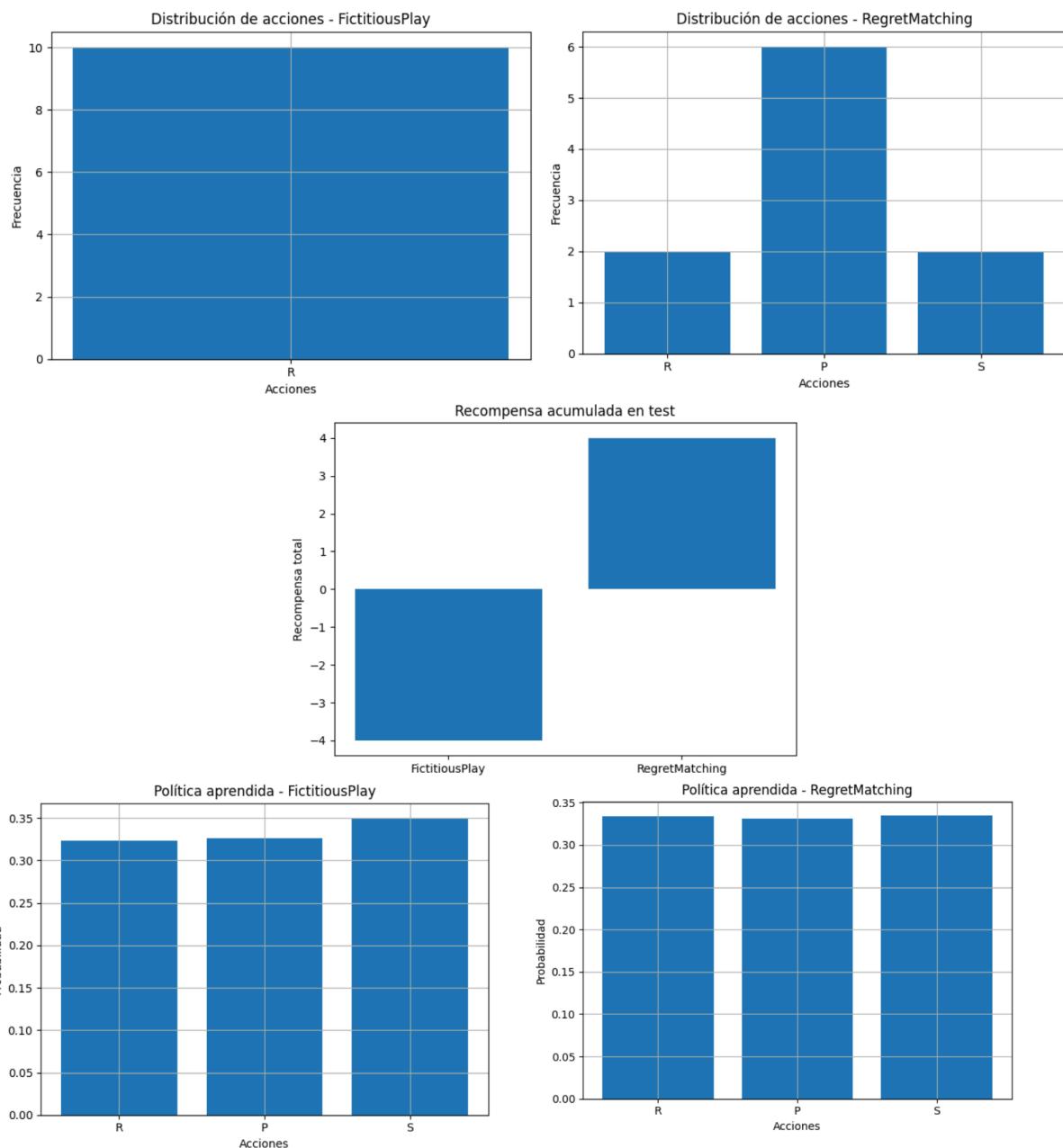
En un segundo test, vemos un comportamiento exactamente igual. RM define una política igual a la de RM pero ejecuta una única acción y acaba perdiendo frente a RA.



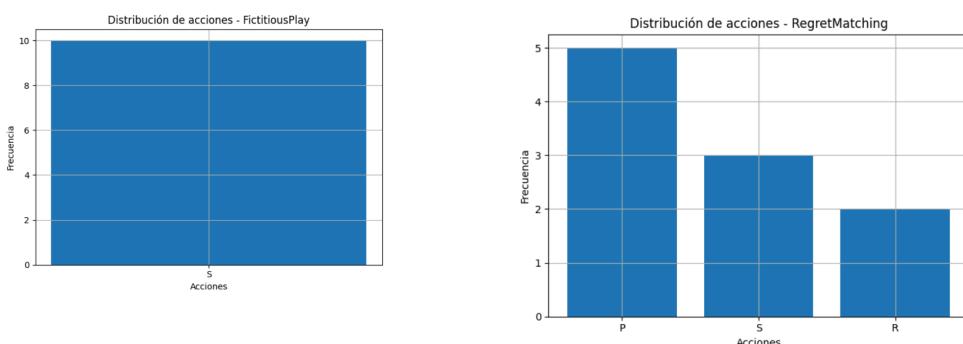


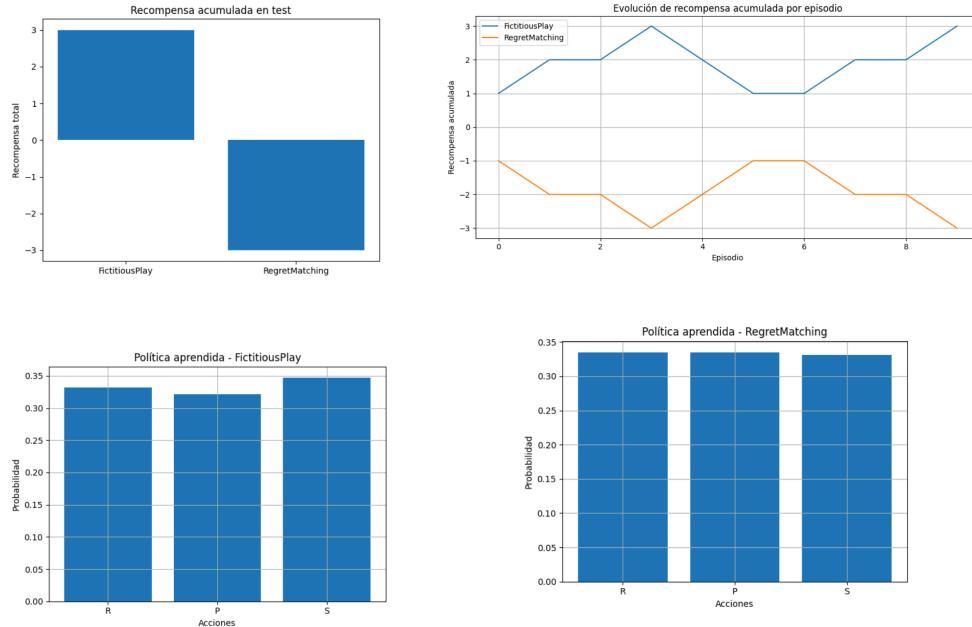
FP vs RM en RPS

Ahora bien, cuando enfrentamos RM contra FP en RPS, vemos un escenario similar a lo expuesto en los dos anteriores. FP aprende un set de políticas donde una de las acciones tiene mayor probabilidad de ser seleccionada, pero con altas chances de que cualquiera de las dos otras sean seleccionadas por igual. En cambio, RM genera un set de posibles movimientos similar a como actuaría RA. Es decir, todas las acciones tienen la misma probabilidad de ser seleccionadas. Al igual que en el primer escenario, es el agente que selecciona una diversidad mayor de acciones quien termina obteniendo un mayor reward acumulado, en este caso RM.



En la segunda prueba, vemos un comportamiento igual a la primera ejecución. La diferencia en este caso es que FP es quien termina venciendo a RM. A igual política, obtenemos resultados diferentes al tener una distribución de acciones diferentes.





Blotto

Este es el juego que nos permite experimentar de diferentes formas pues existen mayor variedad por las posibles combinaciones existentes. Las combinaciones posibles son determinadas en el ambiente antes de comenzar el juego y por ello haremos tres tipos de juegos de Blotto: (5, 2); (7, 3) y (15, 5). Este juego contrapone dos distribuciones de un número determinado de soldados en una cantidad de ranuras.

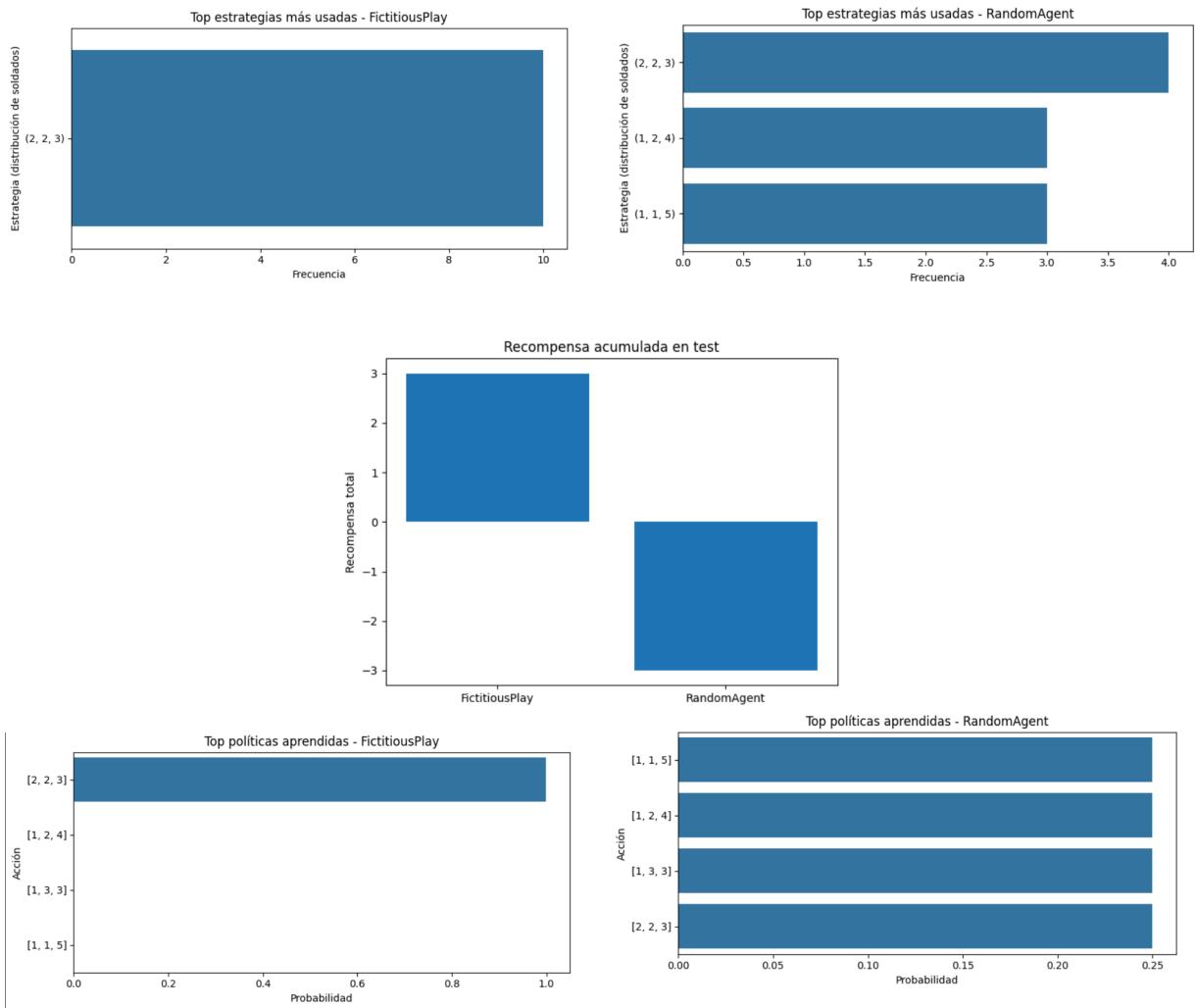
Blotto(5, 2)

Para este ambiente la posibilidad de acciones es acotada y rápidamente cada cruce llega a un equilibrio donde ninguno de los agentes supera al otro. La recompensa acumulada de cada uno de los agentes, para cada uno de los escenarios es igual a cero. Por ese motivo, es necesario utilizar ambientes más amplios para aumentar el rango de acciones y tomar mejores observaciones.

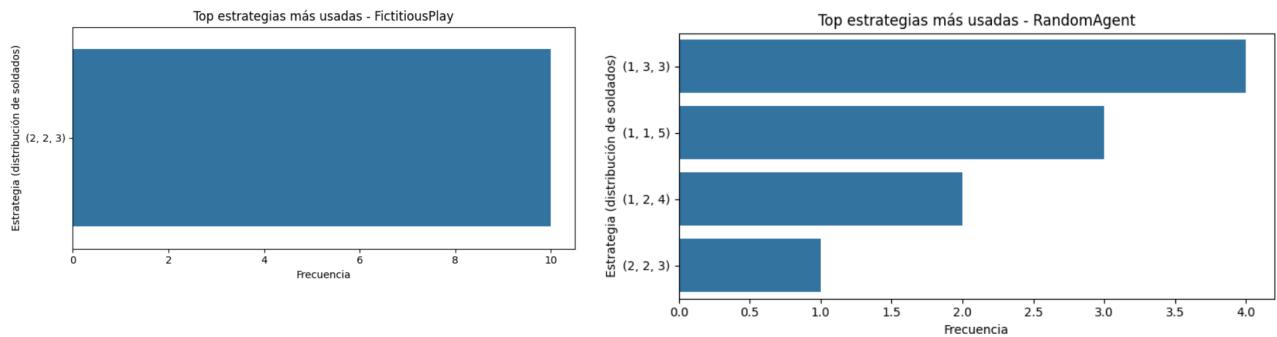
Blotto(7, 3)

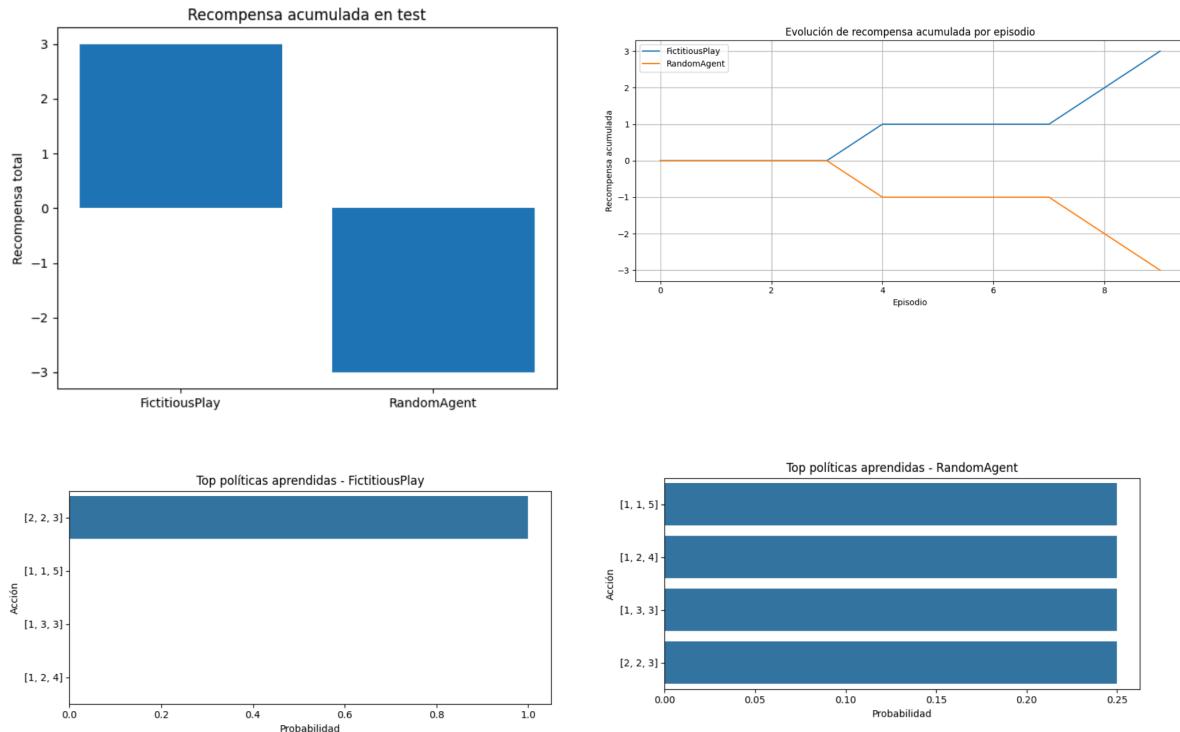
FP vs RA en Blotto(7, 3)

Para este ambiente, FP consigue vencer a RA. Al tomar siempre la mejor acción aprendida, elige la combinación que tiene mayores chances de retornar un reward positivo o al menos igual a cero. Esta es (2, 2, 3). En cambio, RA alterna entre otras combinaciones no competitivas y causa que sus chances de ganar sean menores.



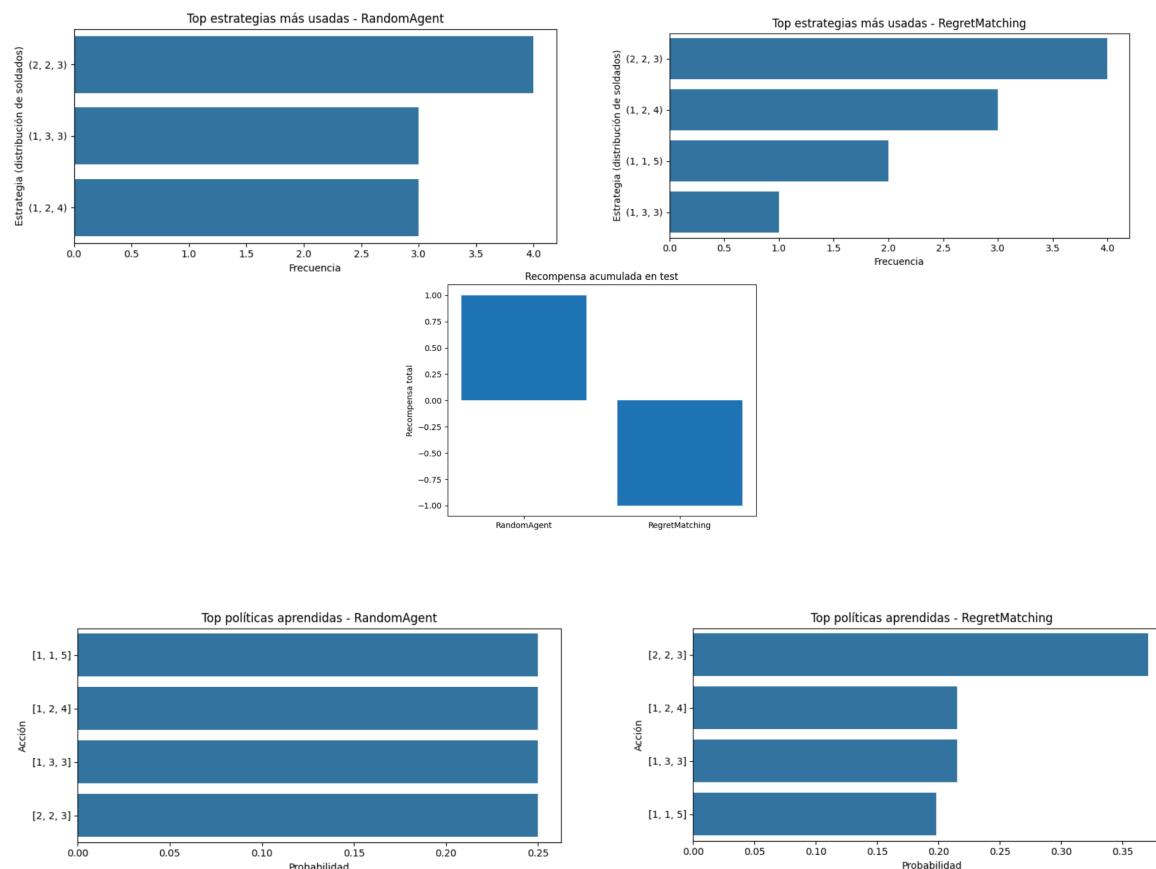
En una segunda ejecución vemos un comportamiento igual. FP aprende una política de una única acción, RA al tener demasiadas acciones a ser tomadas, fluctua y termina perdiendo ante FP.





RM vs RA en Blotto(7, 3)

El cruce entre RM y RA en este ambiente genera como ganador a RA. RA itera por igual entre diferentes combinaciones mientras que RM tiene dificultades para definir una estrategia óptima para la cual ceñirse. Al parecer este ambiente continúa siendo demasiado escueto como para que RM consiga sobreponerse ante RA.

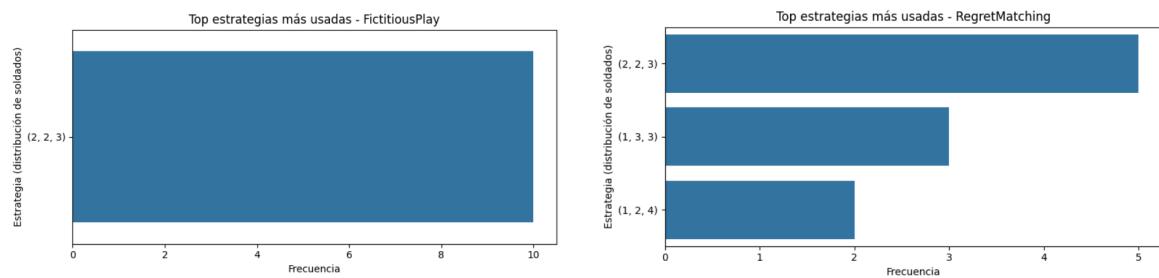


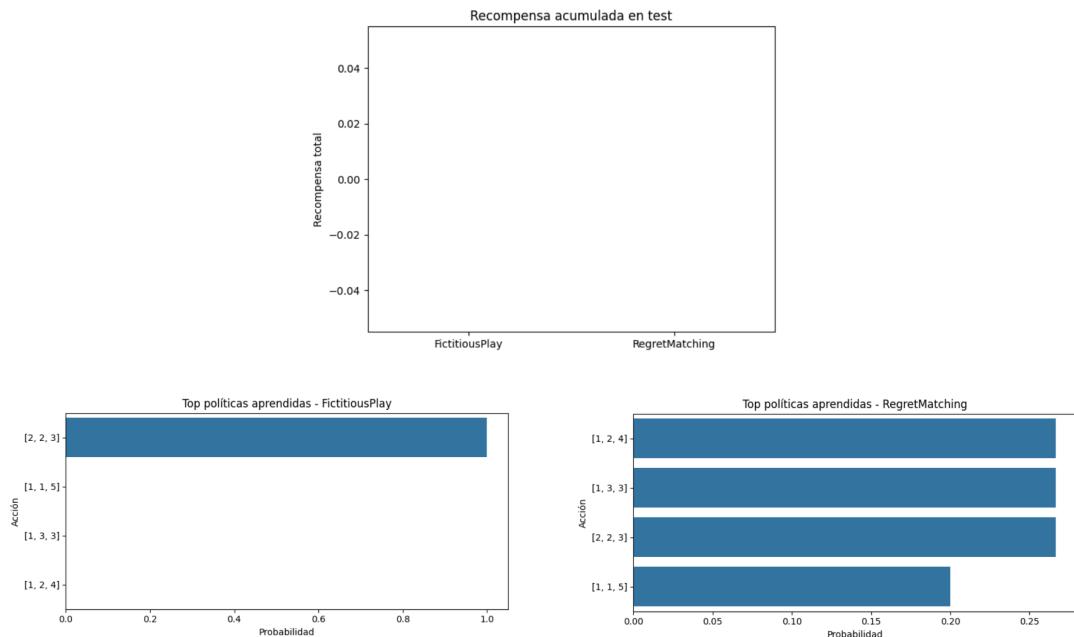
En una segunda prueba, vemos un comportamiento similar de ambos agentes, solo que esta vez es RM quien termina vencedor.



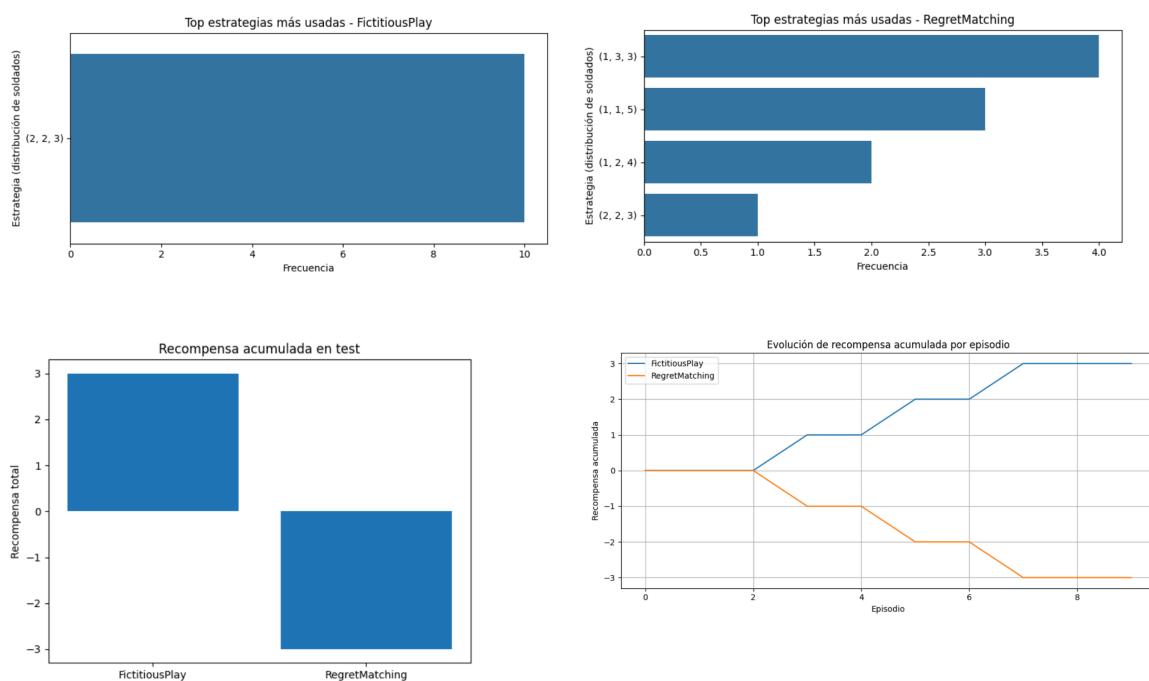
FP vs RM en Blotto(7, 3)

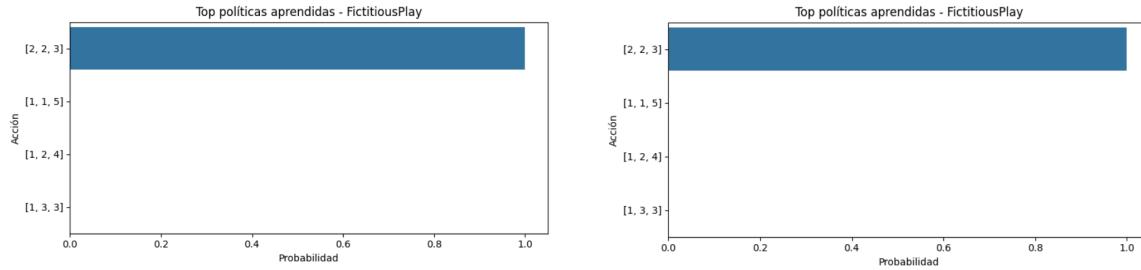
El juego entre FP y RM lleva a un equilibrio de Nash. RM aprende una estrategia mixta mientras que FP aprende una estrategia pura. En este ambiente definido con las diferentes políticas aprendidas por cada agente, no se consiguen superar uno a otro esencialmente.





Al ejecutar nuevamente la prueba vemos que RM comienza a oscilar entre acciones a tomar, y al realizarlo pierde una y otra vez. De este modo, el reward acumulado se inclina hacia FP.

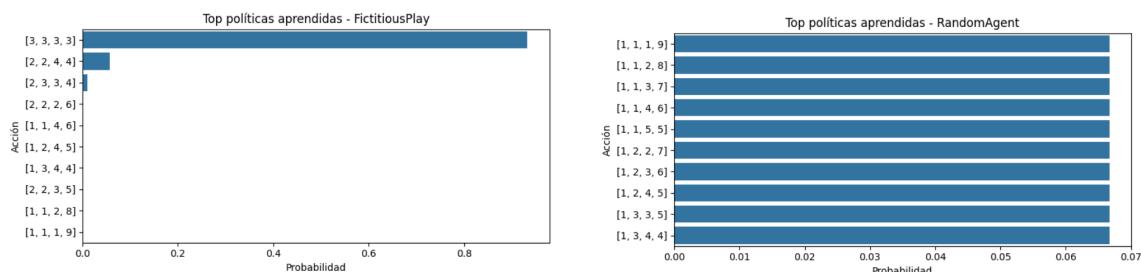
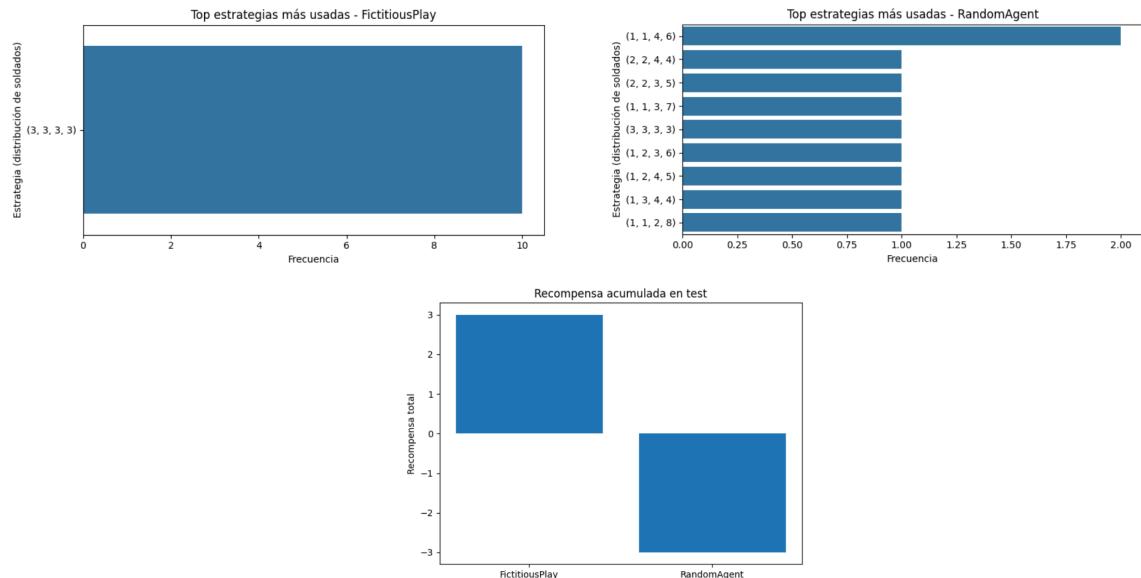




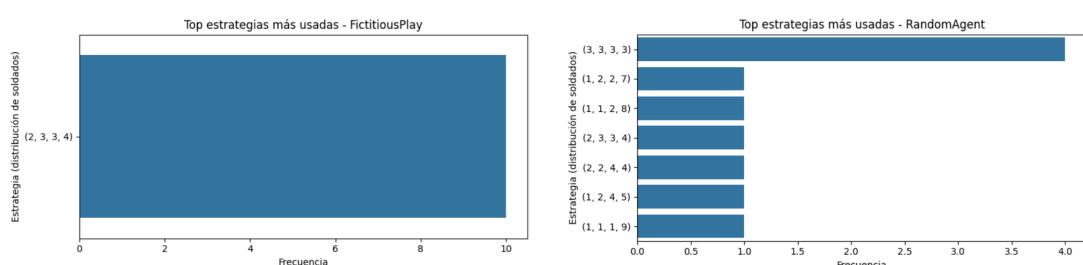
Blotto(12, 4)

FP vs RA en Blotto(12, 4)

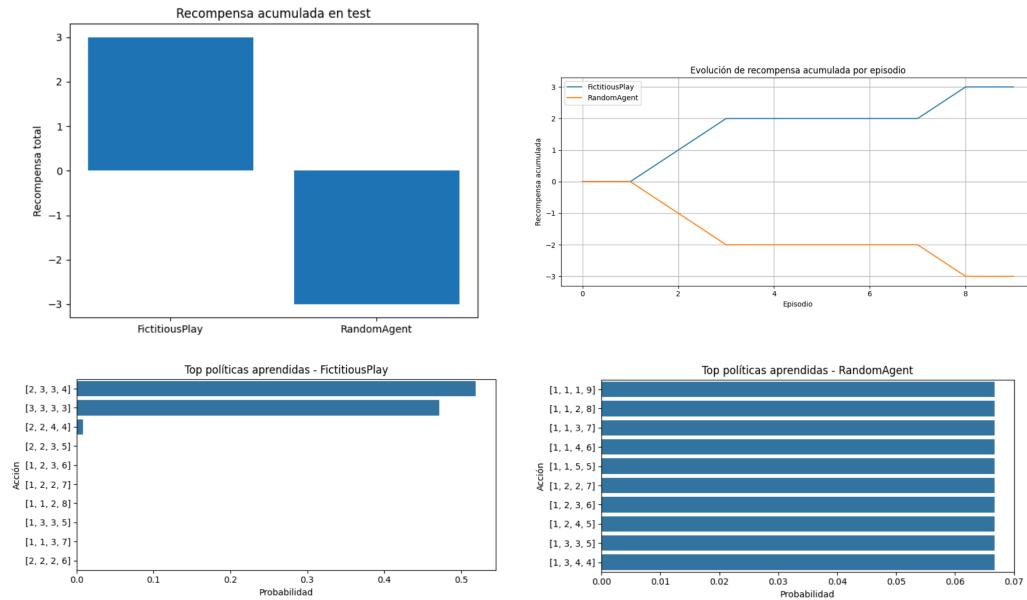
Nuevamente, FP vence a RA. FP logra identificar la mejor combinación posible y supera la política aleatoria de RA con una estrategia pura. No se observan grandes cambios respecto a escenarios anteriores, salvo que en este entorno más amplio FP contempla la posibilidad de jugar diferentes combinaciones distintas aunque con mucho menor posibilidad de jugarlas.



En una segunda prueba observamos un comportamiento muy similar. La única diferencia es que FP no adquiere una estrategia pura, aunque no varía mucho sus acciones y su

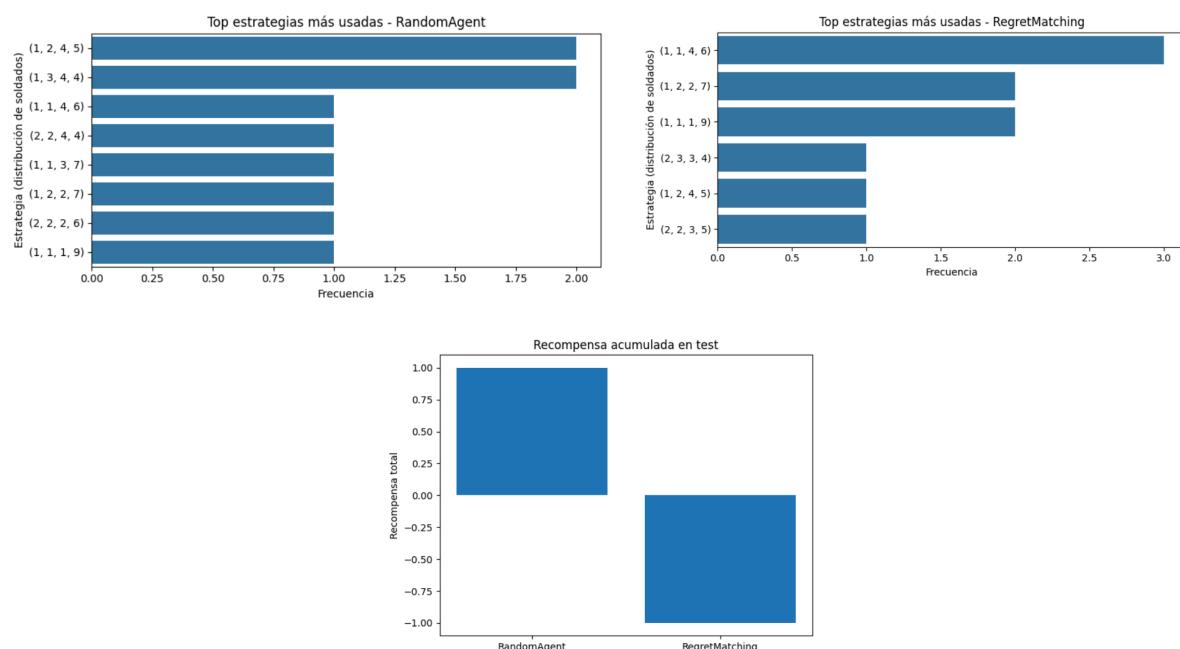


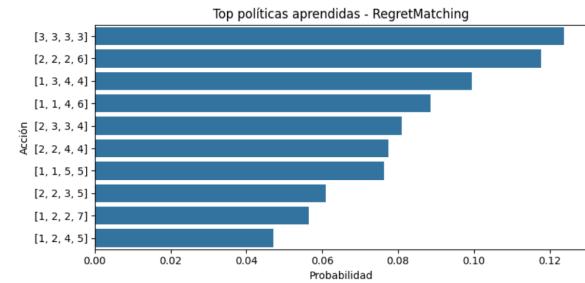
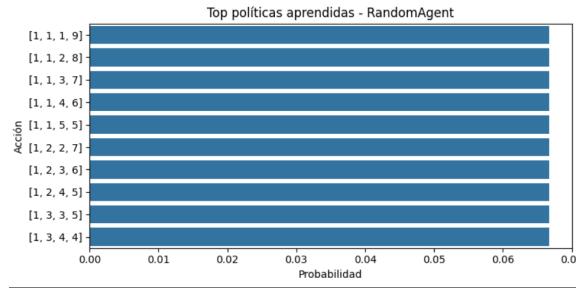
distribución de probabilidades está muy concentrada en una única acción. Nuevamente, FP efectúa la misma acción la mayor parte del tiempo y gana sistemáticamente ante RA.



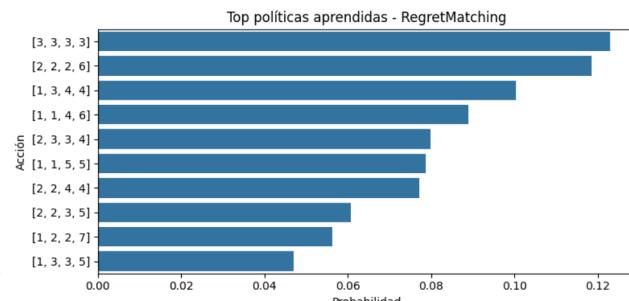
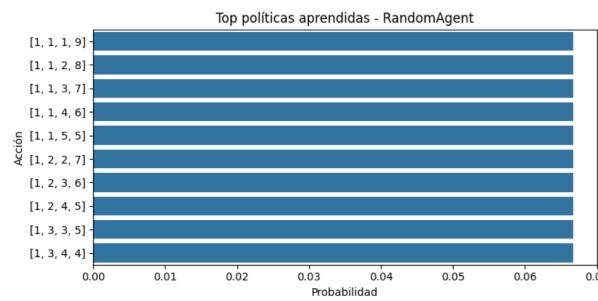
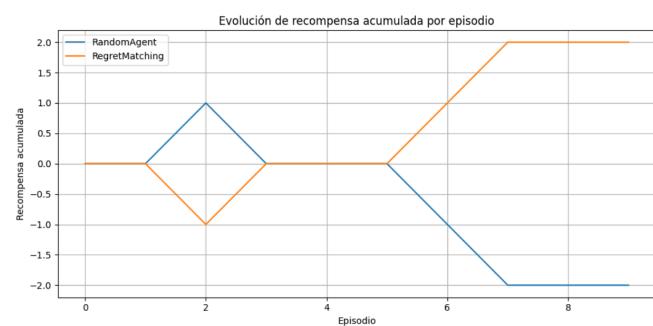
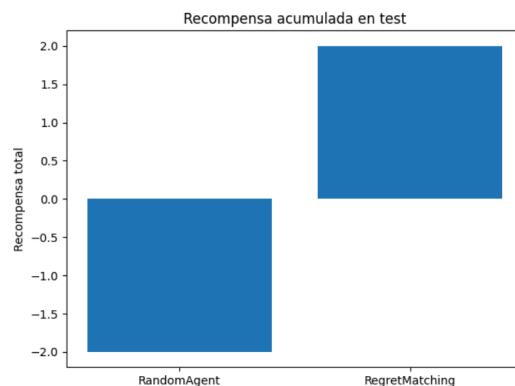
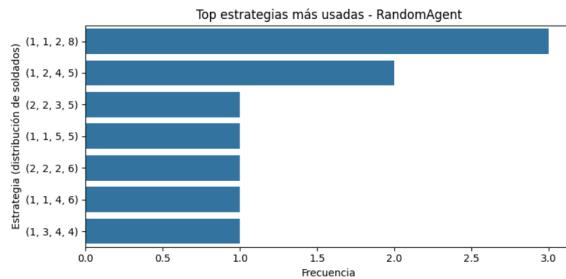
RM vs RA en Blotto(12, 4)

Por su parte, en RM contra RA, RM aprende una política con distribución más uniforme, con una mayor variedad de acciones posibles. Durante el proceso de test, generamos 10 únicos episodios y RM no consigue superar a RA. Es posible que para este caso se pueda mejorar la política del agente RM ajustando sus hiperparametros.



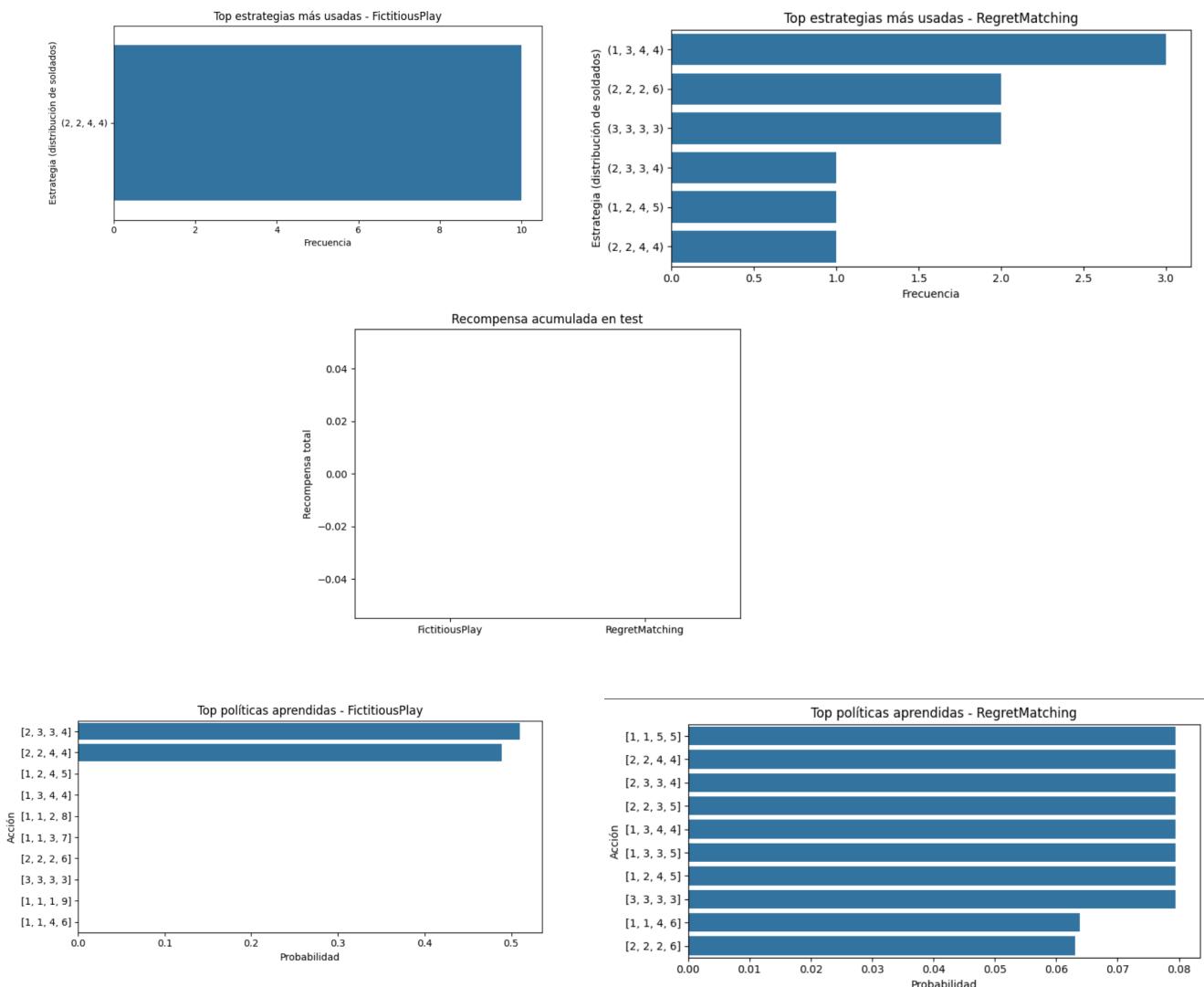


En nuestro segundo test, RM consigue vencer a RA. En este caso, RM realiza una única acción durante las pruebas y esto logra que el reward acumulado sea superior al de RA.

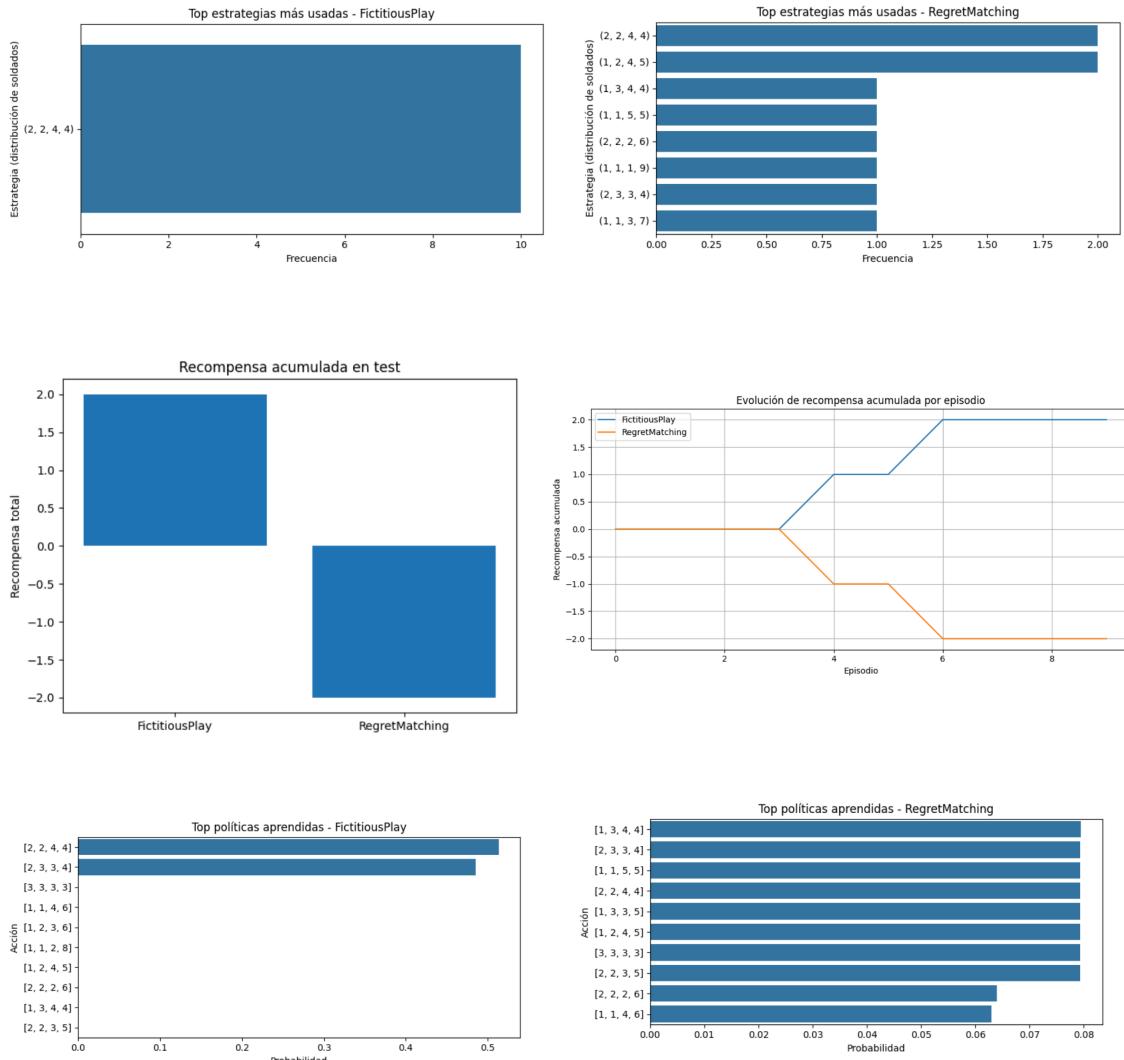


FP vs RM en Blotto(12, 4)

En el enfrentamiento entre RM y FP se alcanza un equilibrio de Nash aproximado, donde indica que sus políticas aprendidas se contrarrestan mutuamente y ninguno obtiene un *reward* acumulado distinto de cero. FP aprende dos combinaciones de acciones óptimas, mientras que RM mantiene una diversidad de elecciones. Esta combinación de estrategias convergen a un equilibrio de Nash.



En nuestra segunda prueba, vimos un resultado diferente. Ambos agentes no consiguen llegar a un equilibrio, sino que FP es quien vence en el juego. Las distribuciones de las acciones fueron muy similares para esta segunda prueba. Mientras que las acciones tomadas por cada agente, RM varió más, en las combinaciones de acciones, que en la primera prueba.



Segunda Parte: Stochastic Games

Los tipos de juegos estocásticos considerados en este estudio son totalmente observables, donde todos los agentes tienen acceso a todos los datos disponibles. En particular se implementan dos modelos (Independent Q Learning y Joint Action Learning con Agent Modeling), sobre el ambiente foraging.

Foraging

Para esta parte se experimenta con distintas configuraciones del juego Foraging con un mismo conjunto de agentes Independent Q Learning.

Las configuraciones todas tienen un tablero de 5x5 y 1 manzana. Se cambia la cantidad de jugadores y la modalidad cooperativo. El juego consiste en recoger las manzanas con los movimientos arriba, abajo, izquierda, derecha, recoger y noop (no hacer nada), sumando recompensas por cada manzana obtenida.

Dimensiones	Jugadores	Agentes	Manzanas	Coop	Iteraciones	Episodios
5x5	2	IQL-IQL	1	No	100	50
5x5	3	IQL-IQL-IQL	1	No	100	100
5x5	3	IQL-IQL-IQL	1	Si	100	100
8x8	2	IQL-IQL	1	No	100	60
8x8	2	IQL-JAL	1	No	100	120
5x5	2	JAL-JAL	1	No	100	80
5x5	2	IQL-JAL	1	No	100	80
5x5	3	IQL-IQL-JAL	1	No	100	100
5x5	3	IQL-JAL-JAL	1	No	100	100
5x5	3	JAL-JAL-JAL	1	No	100	100
5x5	2	IQL-JAL	1	Si	1000	100
5x5	2	JAL-JAL	1	Si	1000	100
5x5	3	IQL-JAL-JAL	1	Si	100	100
5x5	3	IQL-IQL-JAL	1	Si	100	100

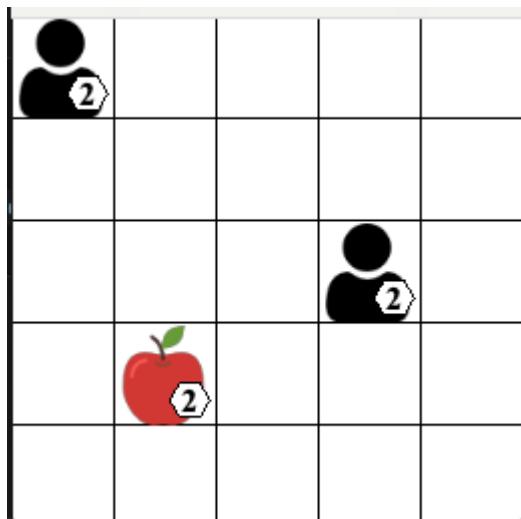
Aumentar el tamaño del tablero aumenta el costo computacional, ya que hay más estados para recorrer, así como la cantidad de jugadores.

A diferencia de la parte anterior, donde se pueden garantizar resultados bajo ciertas circunstancias, en juegos estocásticos no hay garantías.

Para esta parte también se utilizó una semilla (random seed) para que al inicializar las posiciones sean siempre las mismas mientras dicha semilla no se cambie.

A continuación se presentan las observaciones del tablero por parte de cada agente, y gráficas de la evolución de los rewards promedio acumulados durante el entrenamiento.

La siguiente figura es un render del juego en un cierto estado. Permite visualizar mejor el comportamiento de los agentes.



Para el primer caso de la tabla, el juego se inicializa con las siguientes observaciones:

Agent: agent_0

Observation: [3. 1. 2. 2. 3. 2. 0. 0. 2.]

Agent: agent_1

Observation: [3. 1. 2. 0. 0. 2. 2. 3. 2.]

Estos vectores indican la posición de las manzanas y los jugadores. Están agrupados de a ternas, donde los primeros 2 números de cada terna indican la posición de la manzana y de los jugadores, y el último dígito es una flag que indica si existe, o si fue recogida en el caso de la manzana o si están visibles en el caso de los agentes, mostrando la manzana primero y el agente que hace la observación en segundo lugar.

Independent Q-Learning

El primer algoritmo presentado en esta parte. Es el más sencillo de los dos, ya que depende únicamente de sus acciones y sus recompensas observadas.

Juegos competitivos de IQL

Foraging 5x5 2v2: IQL vs IQL



Primera configuración en la tabla, la más simple. Se observa una estrategia dominante de parte del agente 0.

Foraging 5x5 2v2: IQL vs IQL vs IQL

Para el segundo caso se tiene:

Agent: agent_0

Observation: [3. 1. 2. 0. 0. 2. 4. 1. 1. 4. 2. 1.]

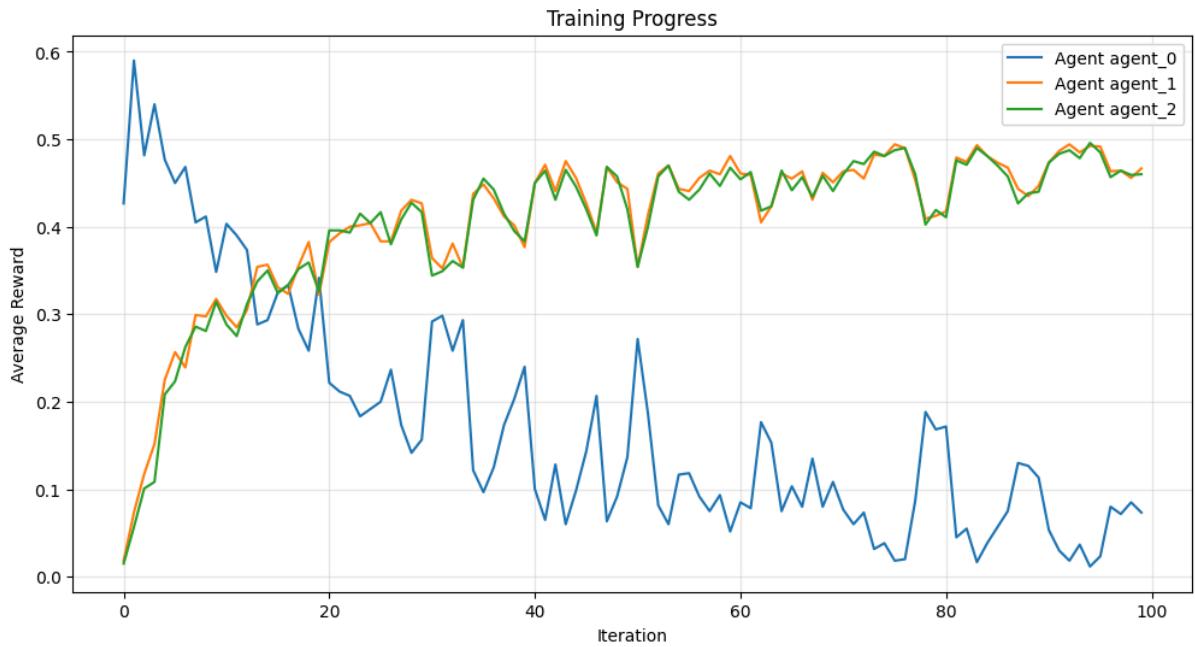
Agent: agent_1

Observation: [3. 1. 2. 4. 1. 1. 0. 0. 2. 4. 2. 1.]

Agent: agent_2

Observation: [3. 1. 2. 4. 2. 1. 0. 0. 2. 4. 1. 1.]

con el siguiente resultado.



En la siguiente gráfica de recompensas se observa que los agentes 1 y 2 adquieren comportamientos muy similares, al mismo tiempo que obtienen una mejor estrategia. Cabe destacar que en el estado inicial, los agentes 1 y 2 están cerca en el tablero.

Foraging 8x8 2v2: IQL vs IQL

En la cuarta configuración los agentes inicializan de la siguiente manera:

Agent: agent_0

Observation: [6. 2. 2. 4. 6. 2. 0. 1. 2.]

Agent: agent_1

Observation: [6. 2. 2. 0. 1. 2. 4. 6. 2.]



Esta configuración es sencilla en el sentido de que tiene 2 jugadores, pero el tablero es más grande, lo cual da lugar a una mayor cantidad de estados posibles. Se observa una dominancia del agente 0 pero finalmente llegan a un equilibrio.

Juegos cooperativos de IQL

Foraging 5x5 2v2 Coop: IQL vs IQL vs IQL

El tercer escenario inicia en el estado

Agent: agent_0

Observation: [3. 1. 4. 0. 0. 2. 4. 1. 1. 4. 2. 1.]

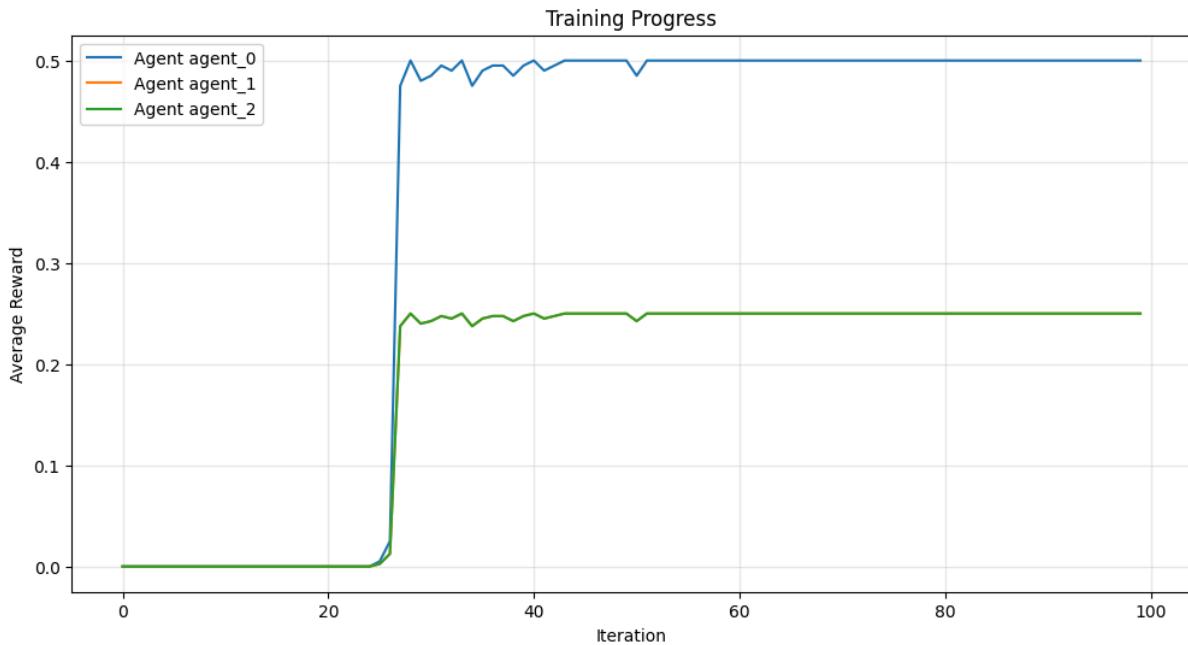
Agent: agent_1

Observation: [3. 1. 4. 4. 1. 1. 0. 0. 2. 4. 2. 1.]

Agent: agent_2

Observation: [3. 1. 4. 4. 2. 1. 0. 0. 2. 4. 1. 1.]

y se obtiene el siguiente resultado:



Esta configuración es en modalidad cooperativa, a diferencia de las anteriores. Se puede observar que a partir de cierto punto se obtienen comportamientos bastante estables, aunque es necesario entrenarlo con una mayor cantidad de episodios que las otras configuraciones, incluso con dimensiones reducidas en el tablero.

Joint Action Learning - Agent Modeling

El otro algoritmo implementado para juegos estocásticos fue Joint Action Learning con Agent Modeling (JAL-AM).

El agente procura observar las acciones conjuntas de los agentes en cada iteración realizada, mientras que, al mismo tiempo, diseña un modelo interno en base a cómo actúa el agente concorrente.

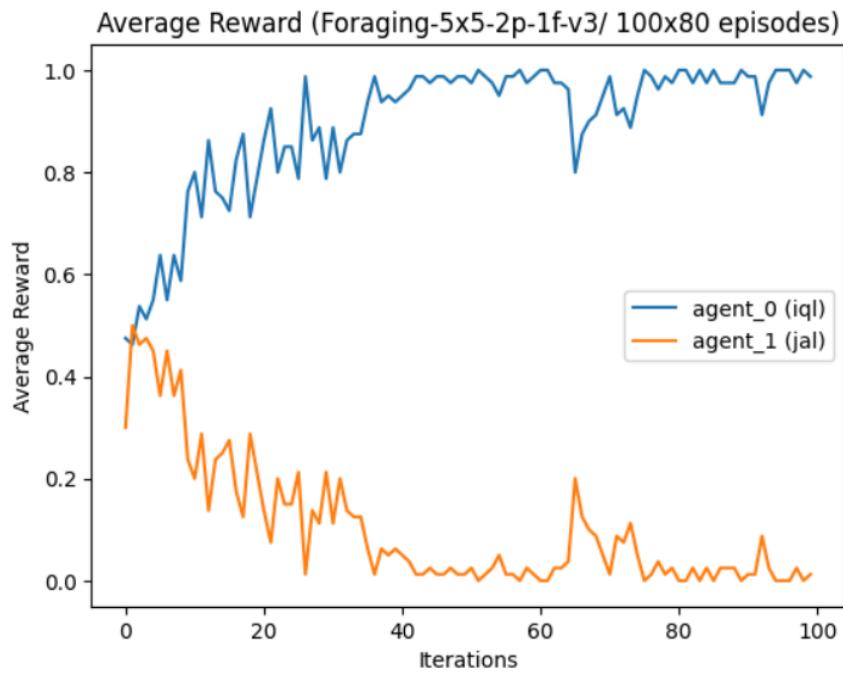
Al igual que IQL, JAL-AM fue utilizado para Foraging para competir contra otros agentes y para cooperar en el mismo juego.

Juegos competitivos de JAL-AM

Foraging 5x5 2v2: IQL vs JAL-AM

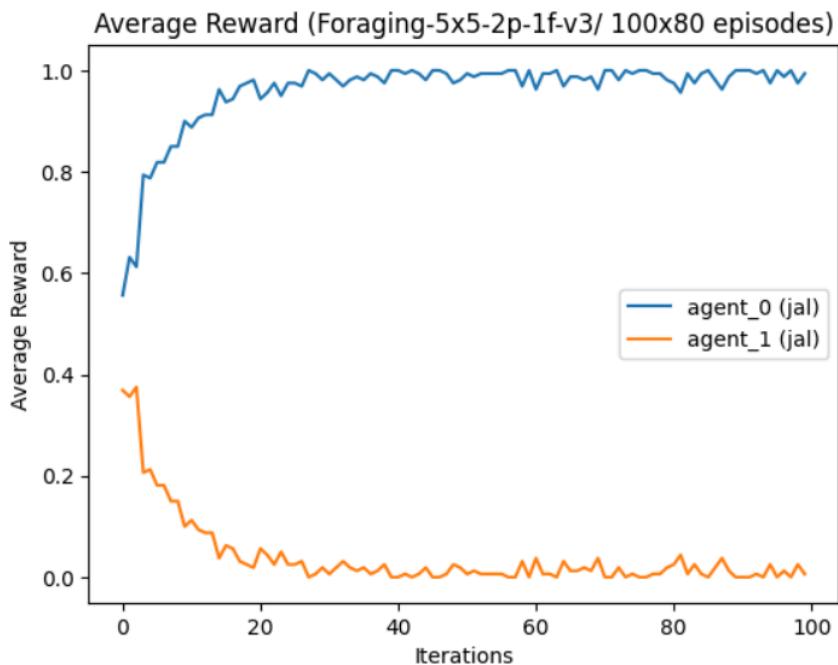
Para los juegos competitivos JAL-AM fue utilizado para juegos de 5x5 contra otro agente y contra otros 2 agentes.

En los juegos competitivos de 5x5 JAL-AM contra IQL, JAL-AM no consigue competir contra su oponente. IQL rápidamente es quien vence y JAL-AM no logra superarlo.



Foraging 5x5 2v2: JAL-AM vs JAL-AM

Una gráfico similar conseguimos generar al enfrentar a JAL-AM contra JAL-AM

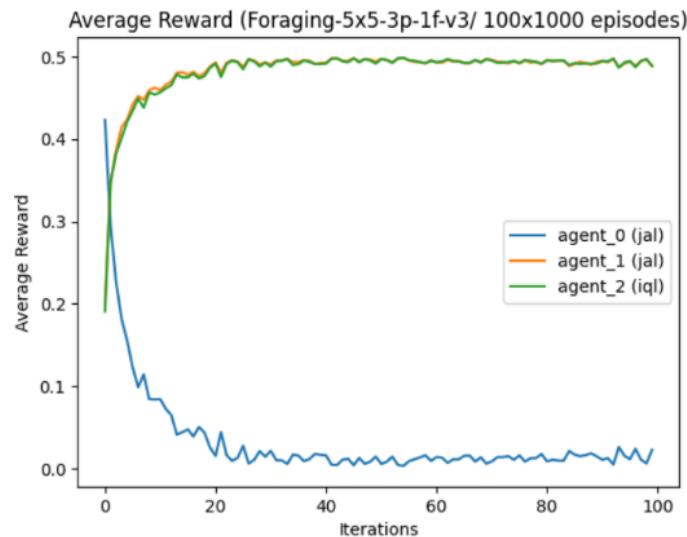


Al ser agentes iguales, podemos observar que el ganador está determinado por la posición inicial de cada agente en el tablero. El agente que al comienzo del juego está más próximo al objetivo, es quién aprende más rápido a cómo alcanzar la manzana e iterar sucesivamente sobre ello.

Foraging 5x5 3v3: IQL vs JAL-AM vs JAL-AM

En nuestro siguiente caso, entre IQL contra dos agentes JAL-AM. Observamos que dos agentes consiguen aprender el camino óptimo hacia la fruta, mientras que uno de los agentes de JAL-AM no consigue tener éxito en aprender a encontrar de mejor forma que sus oponentes la manzana.

Los dos agentes que consiguen aprender y superar la propuesta de juego son IQL y un agente JAL-AM.



Ninguno de estos dos últimos agentes consiguen superarse uno a otro, y el reward acumulado total para cada agente queda en equilibrio.

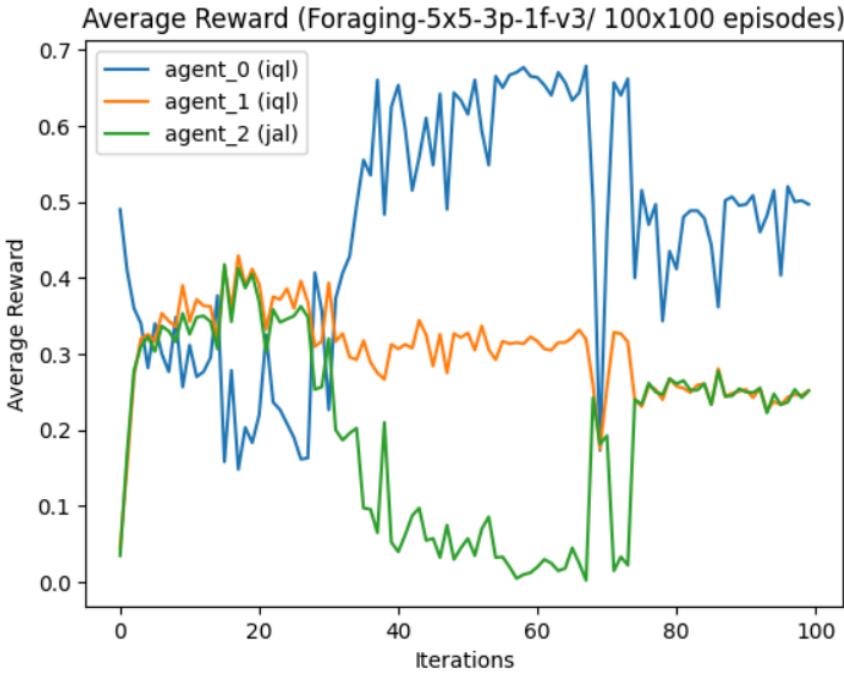
Foraging 5x5 3v3: IQL vs IQL vs JAL-AM

En la propuesta de juego entre dos agentes IQL contra JAL-AM se pueden observar resultados realmente interesantes.

En el primer tramo de las iteraciones, vemos que uno de los agentes IQL domina el juego por poco, mientras que el otro agente IQL encuentra dificultades para superar a sus contrincantes. En cambio, JAL-AM sigue de cerca la performance del primer agente IQL.

En el segundo tramo de las iteraciones, se revierte la situación. El agente IQL que peor perforaba, pasa a estar en primer lugar y así se mantendrá hasta el final de las iteraciones. Mientras que el otro agente IQL se mantiene estable en el tiempo. Por otro lado, JAL-AM parece empezar a tener grandes dificultades para superarse.

En el tramo final de las iteraciones, JAL-AM logra revertir la situación y consigue equilibrar su desempeño con uno de los agentes IQL, el que ha sido más estable durante las iteraciones. En cambio, el agente IQL que mejor performance tenía a medio juego, pasa a decaer en su rendimiento aunque manteniéndose como el número uno.

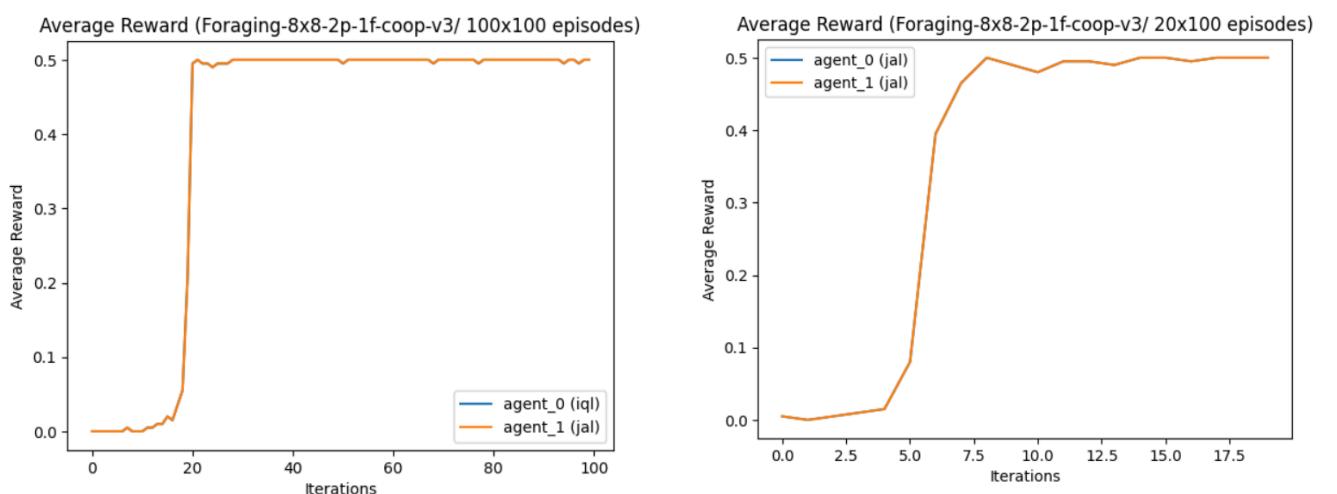


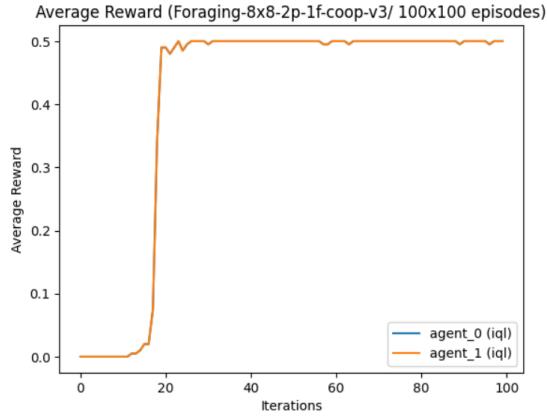
Cabe destacar que en algunos puntos durante las iteraciones, a principio de juego y a medio juego, JAL-AM supera a sus dos agentes contrincantes IQL. No solo eso, sino que cae y vuelve a recomponerse. Una inferencia que podemos realizar al respecto es que JAL-AM podría llegar a tener un mejor rendimiento en caso de tener la suficiente cantidad de iteraciones durante un largo período.

Juegos cooperativos de JAL-AM

Para el caso cooperativo, JAL-AM fue utilizado dentro de diferentes tableros de juegos en conjunto al igual que el agente IQL. Se probaron distintas combinaciones en dos y tres jugadores con IQL y JAL-AM.

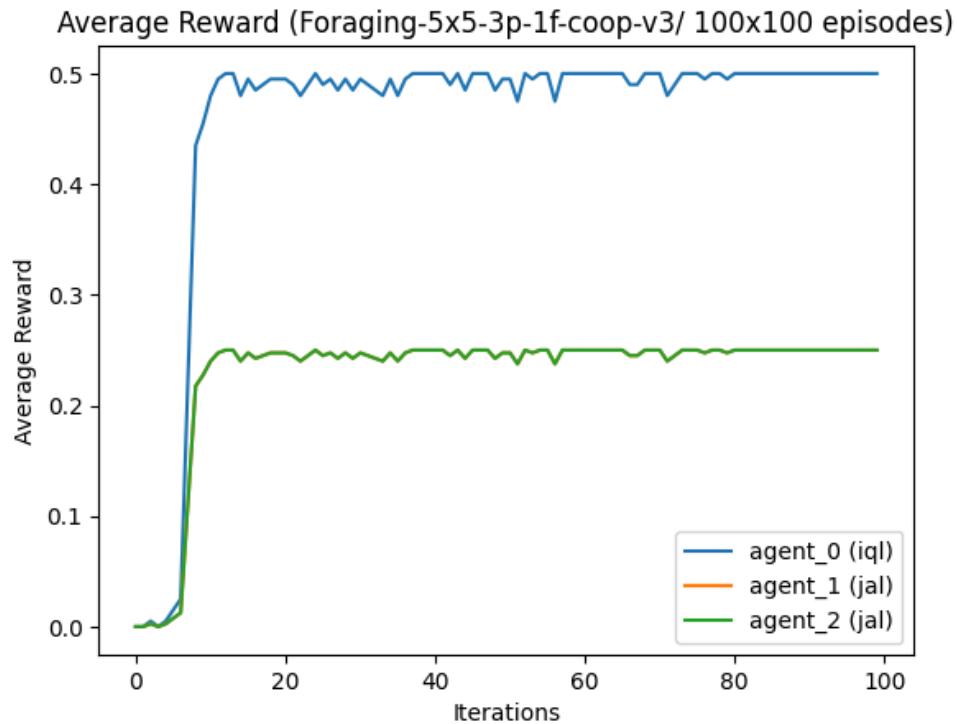
En los escenarios de 2 jugadores, los agentes cooperan y reparten los rewards equitativamente. La diferencia radica en la cantidad de iteraciones que necesita la combinación de agentes para dominar el tablero.





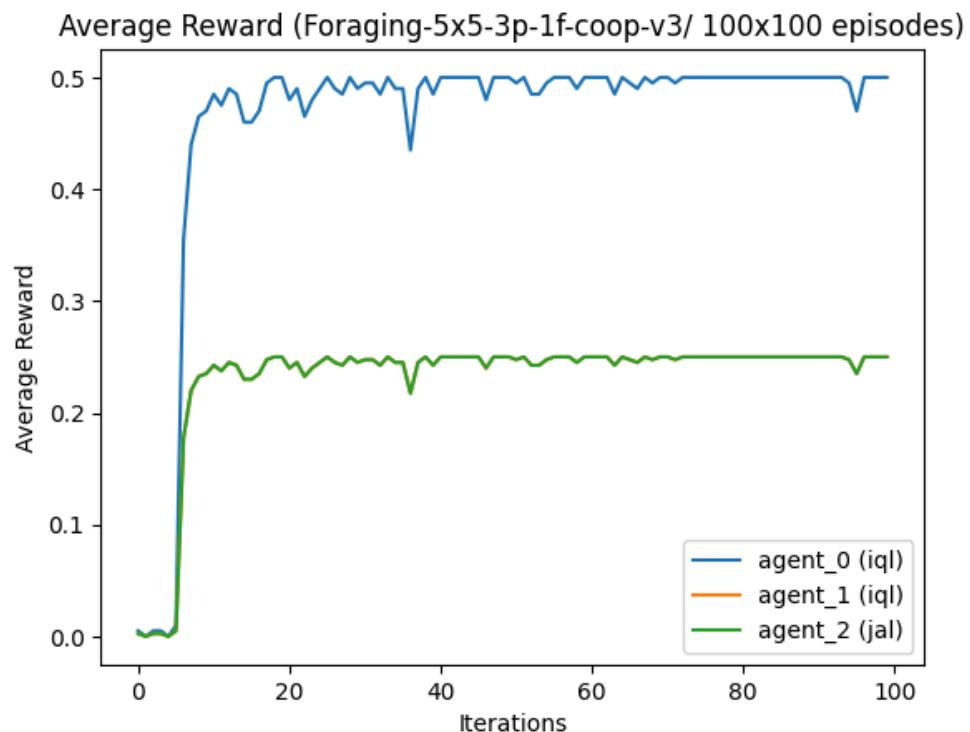
Vemos que para el caso de tener dos agentes JAL-AM cooperando, estos consiguen rápidamente trabajar en conjunto para completar el juego. En cambio, cuando juega un agente IQL con un agente JAL-AM, u otro IQL, estos demoran más iteraciones para dominar el juego. En el primer caso los agentes dominan el juego en menos de 5 iteraciones de 100 episodios. Mientras que para el segundo caso se necesitan de al menos 20 iteraciones.

Foraging 5x5 3v3 Coop: IQL vs JAL-AM vs JAL-AM



En el caso de cooperativo con 3 agentes, se puede observar que cuando se introducen combinaciones donde hay agentes JAL, todos los agentes convergen más rápido a la estrategia resultante, que es similar a la que resulta de tres agentes IQL.

Foraging 5x5 3v3 Coop: IQL vs IQL vs JAL-AM



Esto es consistente con la idea de que JAL-AM es un buen modelo para juegos cooperativos, ya que observa el comportamiento de los demás agentes.

Conclusiones

En relación a los juegos N-forms utilizados: Matching Pennies, Rock, Paper and Scissors y Bloto. Podemos destacar una serie comportamientos comunes a lo largo de los diferentes juegos para cada uno de los agentes utilizados para estos juegos y sus cruces.

El primer comportamiento que observamos es que los agentes muestran un mejor desempeño cuando disponen de un conjunto de acciones posibles para ejecutar, en lugar de estar limitados a una única acción. Esto se evidencia de manera clara en los cruces de RPS, cuando se enfrentan FP con RA o con RM. La flexibilidad para elegir entre diferentes acciones les permite adaptarse mejor a las diversas situaciones que puedan surgir. En particular, en el caso de MP, al enfrentarse FP contra RM, parece que RM seleccionó la respuesta correcta desde el inicio del test y comenzó a ganar repetidamente contra FP. Debido a que FP estaba limitado a una única elección, no tuvo margen para mejorar su desempeño.

En cambio, en juegos más limitados es mejor ceñirse a una única acción y repetirla sistemáticamente. En estos tipos de juegos, FP es quien sale vencedor una y otra vez contra ambos agentes: RA y RM. Estos resultados se repiten una vez que RM acierta rápidamente en la política óptima cuando juega contra RA.

En definitiva, para los N-form games, el agente más óptimo dependerá del tipo de juego.

Los juegos estocásticos superan en complejidad a los matriciales debido a su dependencia temporal, la cual hace más costosa la implementación de los agentes que se pueden entrenar en dichos ambientes.

Podemos afirmar que JAL-AM es un algoritmo más efectivo que IQL, pero a un mayor costo computacional, que se hace notar cuando se aumenta la complejidad del ambiente o la cantidad de agentes que interactúan. También se observa un mejor comportamiento de JAL-AM en juegos cooperativos, lo cual sabemos que es consistente con lo que se ha mencionado sobre cada agente. Las combinaciones de agentes que contienen un JAL-AM consiguen dominar el problema en menos iteraciones que cuando no hay uno de estos agentes.

Para los experimentos realizados tanto entre agentes IQL como entre JAL-AM, y entre ambos, existe una fuerte correlación entre el resultado y las posiciones iniciales de los agentes. Suele ser más probable que termine ganando el agente cercano a la manzana. A medida que las políticas evolucionan, solo reafirman esta idea. Este efecto es más notorio en tableros más pequeños. Al aumentar el tamaño del tablero el efecto de los estados iniciales puede atenuarse.

Por último, otro componente que influye en las políticas aprendidas y que, de cierta forma, impide que los agentes recorran todos los estados es la semilla. Si bien podemos decir que nos permite entrenar de manera consistente y a un costo computacional razonable, esta podría estar sesgando las acciones de los agentes.

Bibliografía

- **Repositorio del proyecto:** LeandroUruguay1994. (2025). *Multiagentes* [Repositorio GitHub]. <https://github.com/leandrouruguay1994/Multiagentes>
- **Albrecht, S. V., Christianos, F., & Schäfer, L. (2024).** *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press.
<https://www.marl-book.com/>