

SID Primavera 2024

Practica 3 (Laboratorio)

Martí Recalde
Arnau Esteban
Oriol Ramos

Mayo/Junio 2024

INDEX

| | |
|---|----|
| 1- ¿Qué parámetros, y con qué valores, hacen que el algoritmo converja mejor en cuanto a recompensa, individual y colectiva? ¿Y en cuanto a tiempo de entrenamiento?..... | 3 |
| 2.-¿Cómo se comportan los agentes cuando están entrenados para optimizar su comportamiento respecto de cada concepto de solución incluido (minimax, Nash, wellness, Pareto)? ¿Para qué tipo de coordinación es más adecuado cada uno (intereses comunes, conflictivos, mixtos) ?..... | 14 |
| 3.- ¿Qué ocurre si entrenamos a dos agentes con dos conceptos de solución diferentes? ¿Son capaces de coordinarse, o de conseguir recompensas individuales independientemente de la política del otro agente?..... | 19 |
| 4.- ¿Cómo se compara JAL-GT, que es un algoritmo multiagente debido a que calcula la Q en base a acciones conjuntas, con IQN (independent QLearning)?..... | 22 |
| 5.-¿Cómo es capaz de generalizar el algoritmo? Es decir: con el resto de parámetros fijados, ¿cómo es capaz el algoritmo de converger a medida que escala el tamaño del entorno (2x2, 3x3, 4x4, 8x8...), el número de agentes (2, 3, 4...), la complejidad de la representación del estado, etc.?..... | 29 |
| 6.-¿Es capaz el algoritmo de generalizar de manera que tenga éxito evaluando mapas que no ha visto en entrenamiento?..... | 35 |

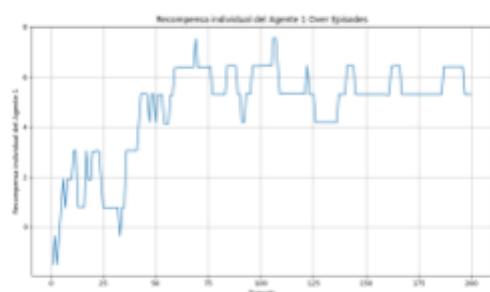
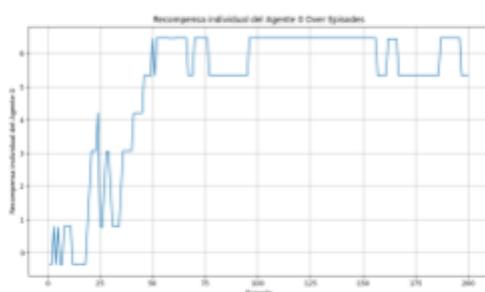
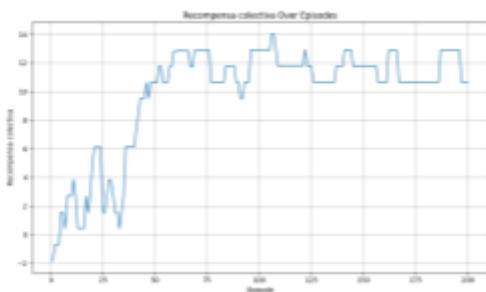
1- ¿Qué parámetros, y con qué valores, hacen que el algoritmo converja mejor en cuanto a recompensa, individual y colectiva? ¿Y en cuanto a tiempo de entrenamiento?

Para atender fidedignamente al propósito de esta pregunta, nos proponemos modificar únicamente los parámetros referentes al algoritmo y su entrenamiento pero no los referentes al entorno. Lo hacemos así por considerar que la variabilidad de convergencia al cambiar el número de agentes, el número de mapas o la densidad de obstáculos no refiere tanto a las cualidades del algoritmo cuanto a la dificultad del problema a tratar. Cuestiones de escalabilidad e incremento de dificultad se analizarán en la pregunta 4.

Nos proponemos modificar, en consecuencia, el número de epochs por ejecución, el número de episodios por epoch, la longitud de los episodios, el learning rate, el valor de epsilon y el concepto solución. No necesariamente en ese orden.

Original:

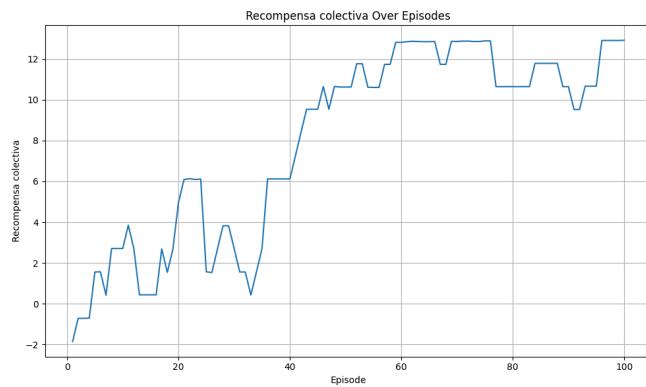
Tiempo de entrenamiento: 1:08



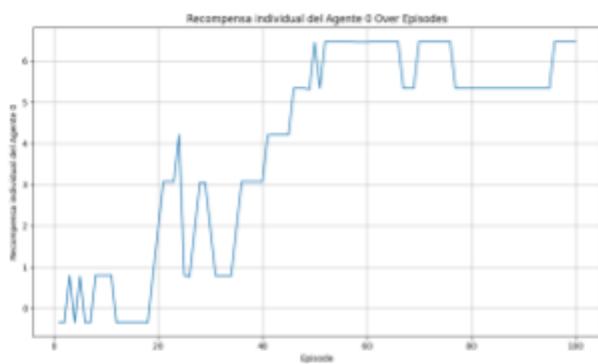
Hipótesis 1: Se puede reducir el número de epochs de 200 a 100 sin afectar al rendimiento de los agentes. Deducimos esto de la observación de las gráficas iniciales en las que, a partir de la epoch 100, los agentes siempre resuelven satisfactoriamente el problema. Hemos escogido modificar este primer parámetro al ver que los agentes convergían a una buena solución con los parámetros iniciales pero que el tiempo de entrenamiento era impráctico para nuestro análisis y se daban muchas iteraciones en que el algoritmo ya había convergido.

Resultados Epoch = 100:

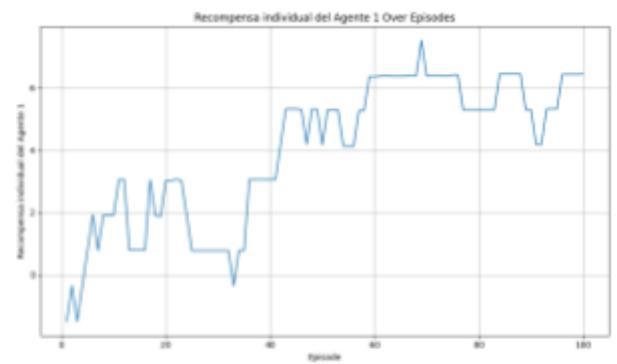
Tiempo de ejecución: 0:35



a) Recompensa colectiva



b) Recompensa individual del Agente 0



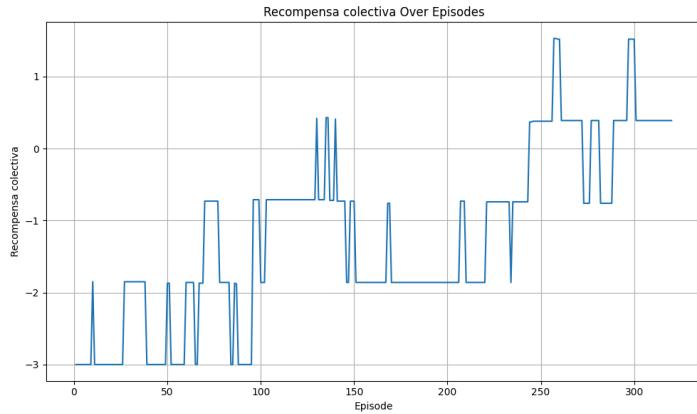
c) Recompensa individual del Agente 1

Como se puede observar, nuestra hipótesis era acertada. Disminuir el número de epochs ha reducido a la mitad el tiempo de entrenamiento sin por ello afectar en lo más mínimo al rendimiento. Seguramente se podría reducir aún un poco más, bajando las epochs en torno a 90, pero no nos parece necesario dado el carácter experimental más que optimizante de nuestra tarea.

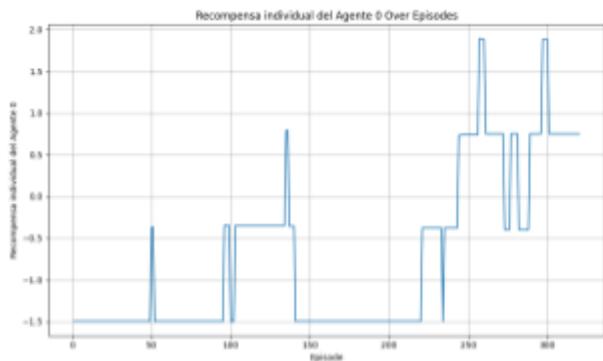
Hipótesis 2: El cambio en el concepto de solución afecta a la convergencia de los algoritmos.

Resultados MiniMax:

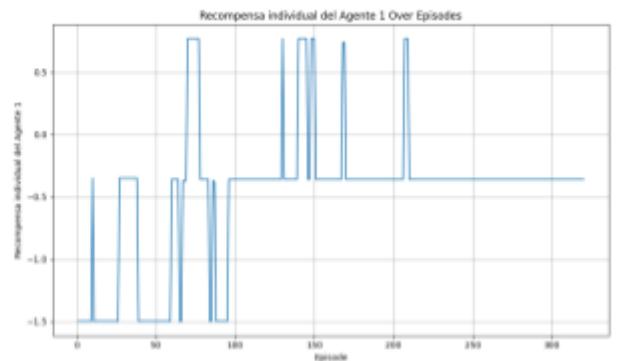
Tiempo de entrenamiento: 1:10



a) Recompensa colectiva



b) Recompensa individual del Agente 0

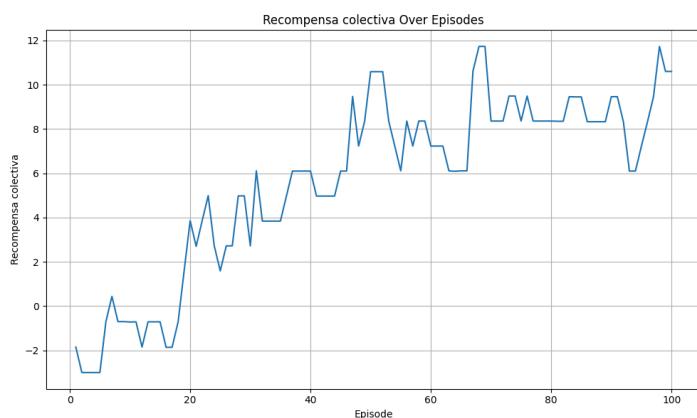


c) Recompensa individual del Agente 1

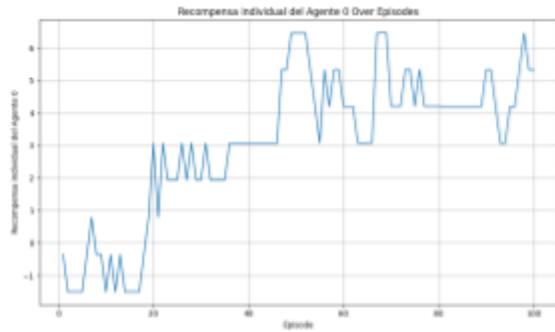
Para usar MiniMax para este problema hemos aumentado el número de epochs hasta 320. Esto se debe a que el entrenamiento con 100 acababa muy rápido pero sin converger. Lo mismo pasaba para 200. Llegados a 320, momento en el que tarda lo mismo en entrenar que el caso original pero no ofrece rendimiento, descartamos esta posibilidad como válida para optimizar la convergencia del algoritmo a ningún respecto.

Resultados Nash:

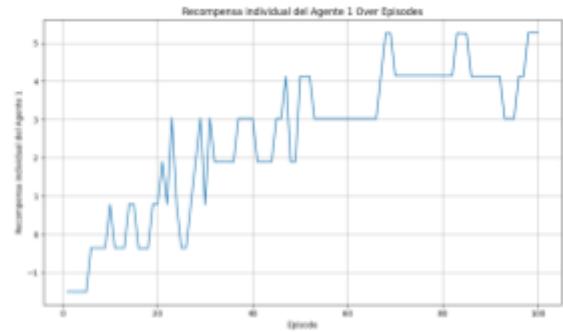
Tiempo de entrenamiento: 0:20 (100 epochs)



a) Recompensa colectiva



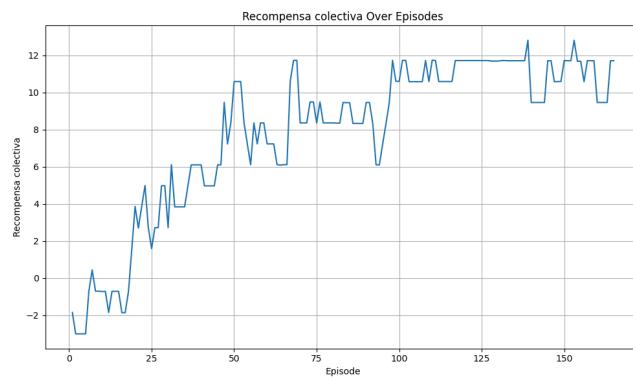
b) Recompensa individual del Agente 0



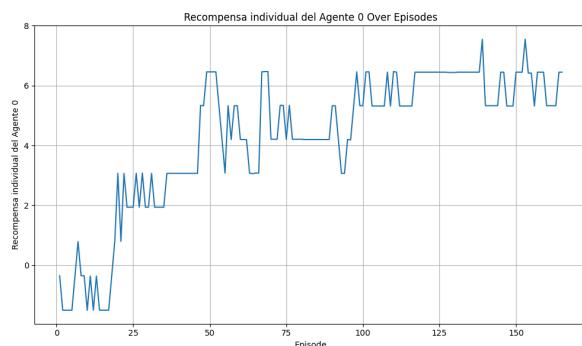
c) Recompensa individual del Agente 1

En vista de que se ejecuta más rápido que Pareto con 100 epochs, pero no converge tan bien, vamos a subir a 165 donde debería tardar lo mismo que esta versión mejor para ver cómo se comporta.

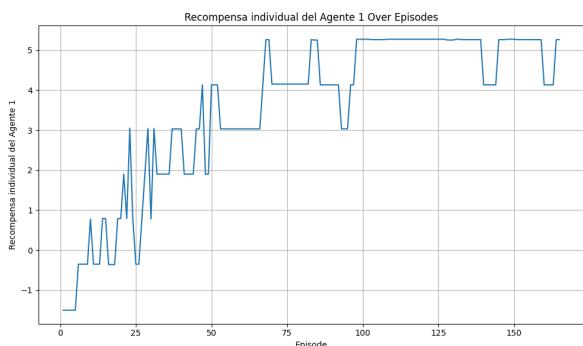
Tiempo de ejecución: 0:31 (165 epochs)



a) Recompensa colectiva



b) Recompensa individual del Agente 0

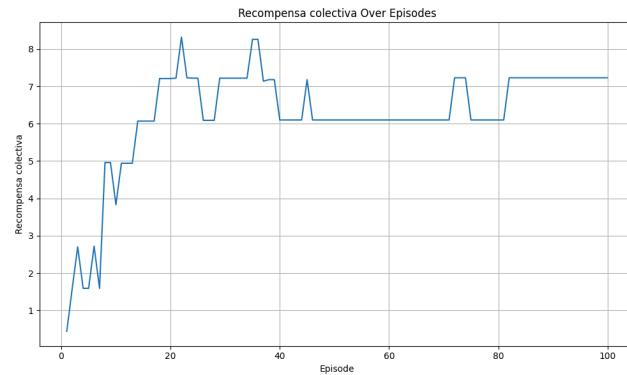


c) Recompensa individual del Agente 1

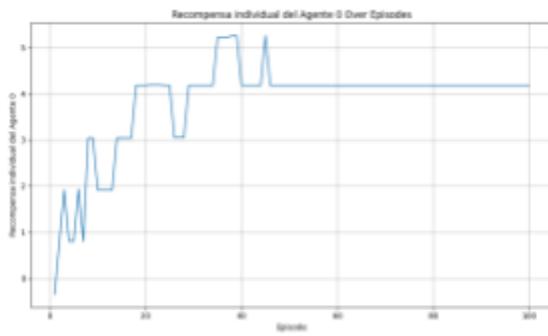
Como vemos, en un tiempo muy similar al de la solución que tomamos por referencia la opción que usa Nash como concepto de solución también parece ser capaz de resolver el problema. No obstante, la convergencia obtenida es mucho más inestable que en el caso con el que la comparamos, siendo así que no sucede que ambos agentes logren estabilizar su capacidad para resolver el problema, por más que los resultados generales obtenidos sean buenos.

Resultados Welfare:

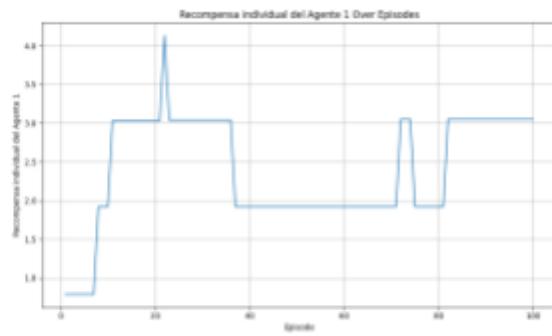
Tiempo de ejecución: 0:20



a) Recompensa colectiva



b) Recompensa individual del Agente 0



c) Recompensa individual del Agente 1

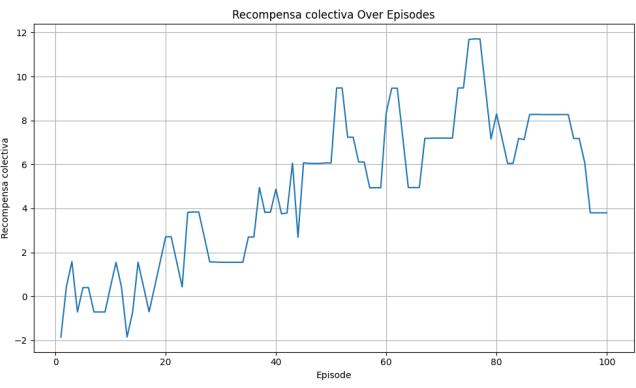
En el caso del concepto de solución Welfare uno de los agentes resuelve permanentemente el problema mientras que el otro no lo hace. Esta opción es más rápida que la que ponemos por referencia pero obtiene peores resultados, por lo que no nos resulta particularmente interesante.

Resultado general 2: El concepto de solución que mejores resultados obtiene en el tiempo mínimo observado para ellos es Pareto. En adelante, seguiremos explorando la modificación de parámetros ciñéndonos a este resultado y al de la hipótesis 1. Volveremos sobre este parámetro en la pregunta 3.

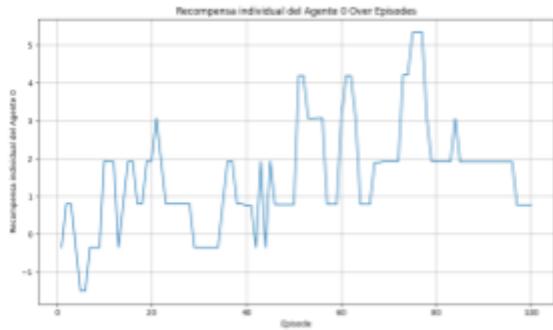
Hipótesis 3: Dado que ahora estamos ejecutando entrenamientos que usan Pareto durante 100 epochs con 10 episodios por epoch, podemos plantearnos reducir el número de episodios por epoch con la esperanza de que siga convergiendo. Esto debería reducir aún más el tiempo de entrenamiento.

Resultados con episodes_per_epoch = 8:

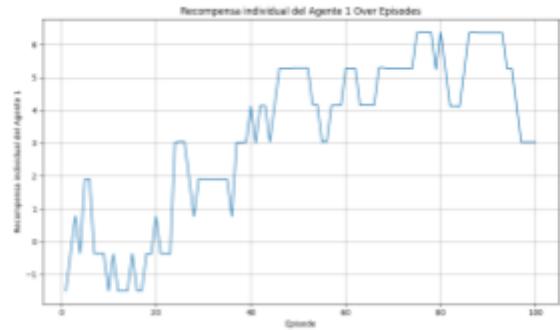
Tiempo de entrenamiento: 0:31



a) Recompensa colectiva



b) Recompensa individual del Agente 0

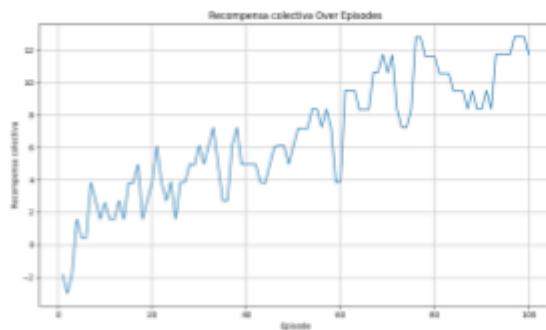


c) Recompensa individual del Agente 1

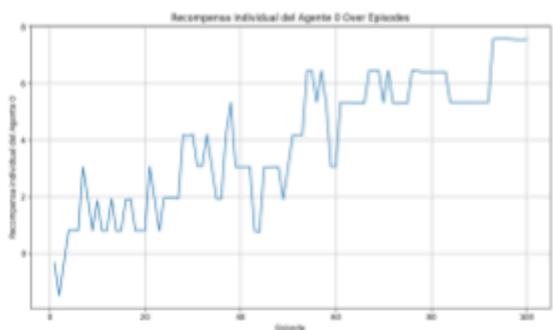
Esta reducción no ha ayudado a mejorar de manera notable el tiempo de entrenamiento (solo hemos reducido 4s) mientras que ha causado una ligera disminución en las recompensas obtenidas tanto conjunta como individualmente. No nos parece un intercambio rentable, por lo que mantenemos nuestro estándar de entrenamiento en los 10 episodios por operación. Probaremos, no obstante, a aumentarlos ligeramente para ver qué sucede.

Resultados con episodes_per_epoch = 12:

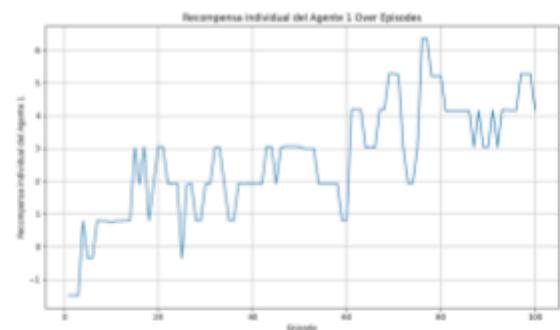
Tiempo de entrenamiento: 0:45



a) Recompensa colectiva



b) Recompensa individual del Agente 0



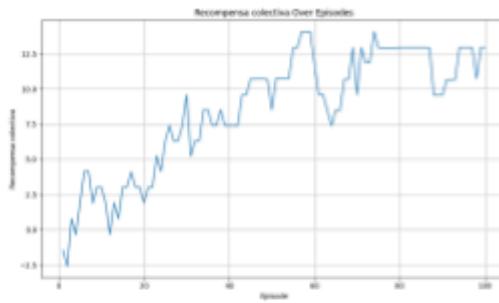
c) Recompensa individual del Agente 1

En este caso, las recompensas obtenidas son tan buenas como las que usamos como referencia si bien es cierto que parece que ambos agentes tardan más en converger hacia las soluciones y que su comportamiento es más inestable. Además, hemos ampliado el tiempo de entrenamiento en un 50% por lo que no parece del todo deseable esta ampliación del número de episodios por epoch. Sería una opción válida para configurar nuestro algoritmo, pues los resultados son buenos, pero nos ceñiremos a la que veníamos usando.

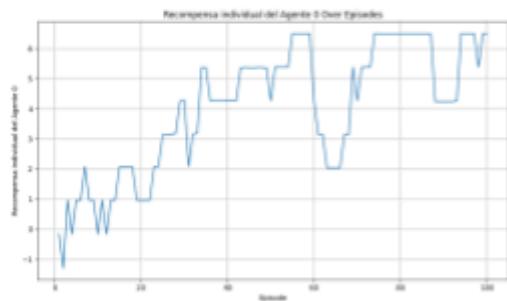
Hipótesis 4: Modificar la longitud de los episodios no tendrá consecuencias positivas en nuestro algoritmo. De reducirla, éste rendirá peor y, en caso contrario, aumentará el tiempo de entrenamiento sin obtener mejores resultados. Esta hipótesis deriva de los resultados de hipótesis anteriores relacionadas con la extensión de los plazos de entrenamiento del algoritmo (ya sea modificando las epochs o los episodios en éstas). El valor original de este parámetro era 16.

Resultados con episode_length = 14:

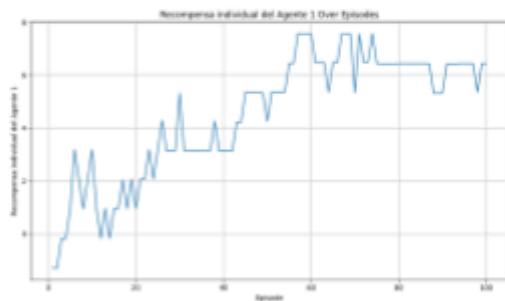
Tiempo de entrenamiento: 0:40



a) Recompensa colectiva



b) Recompensa individual del Agente 0

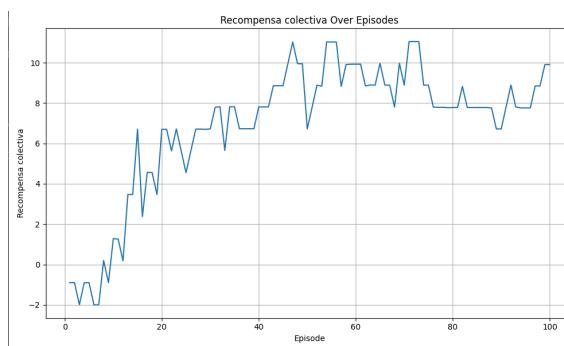


c) Recompensa individual del Agente 1

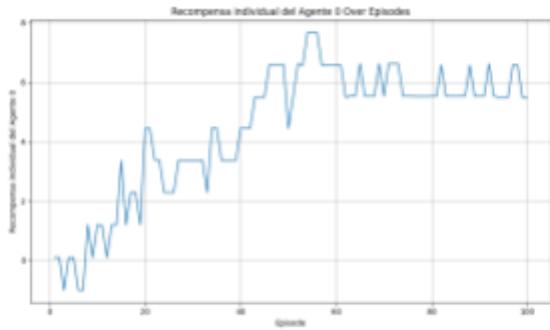
No se observa una gran diferencia con respecto a nuestro estándar. Quizás, a lo sumo, podríamos hablar de una menor estabilización de los agentes en las zonas altas de recompensa. Probaremos una reducción mayor.

Resultados con episode_length = 11:

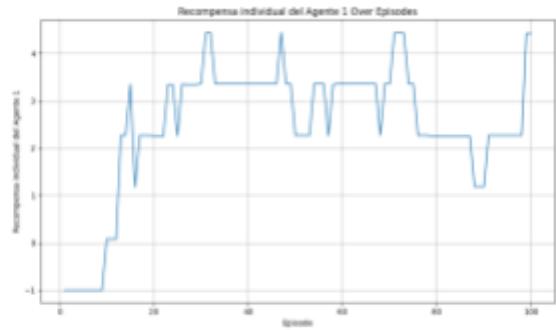
Tiempo de entrenamiento: 0:28



a) Recompensa colectiva



b) Recompensa individual del Agente 0

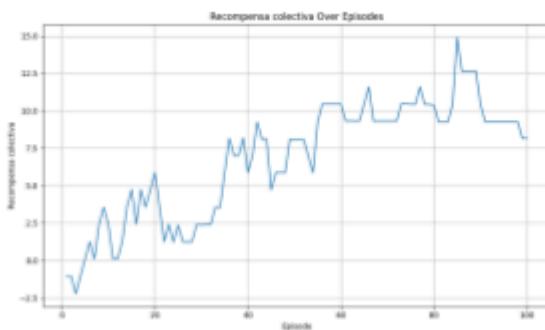


c) Recompensa individual del Agente 1

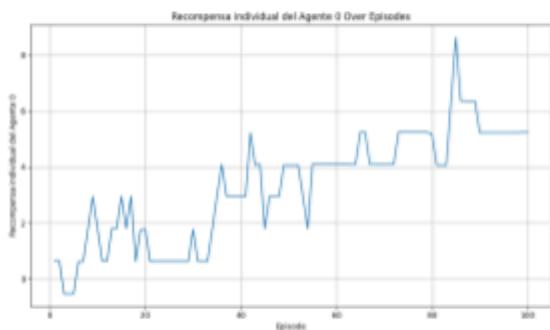
Como esperábamos, hay un ligero deterioro en las recompensas, que no se compensa con los 7 segundos de tiempo de entrenamiento ahorrados. Probaremos ahora a aumentar este parámetro para comprobar la segunda parte de nuestra hipótesis.

Resultados con episode_length = 18:

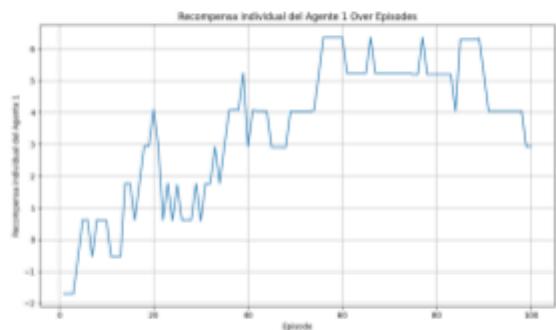
Tiempo de entrenamiento: 0:41



a) Recompensa colectiva



b) Recompensa individual del Agente 0



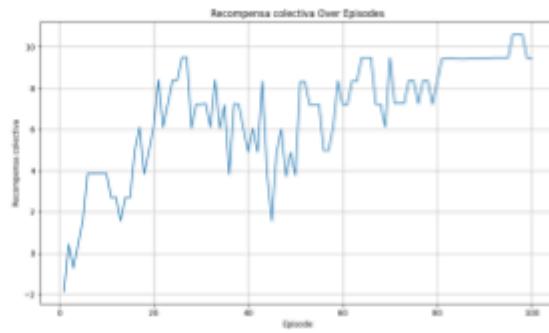
c) Recompensa individual del Agente 1

En esta ocasión, y por primera vez, hemos visto una recompensa colectiva mayor a las de nuestro estándar. Esto, no obstante, no nos parece suficiente para trabajar, a partir de ahora, con esta configuración pues la estabilización de las recompensas obtenidas por los agentes en este caso se da en torno a valores más bajos. Además, un ligero aumento obtenido en lo referente al tiempo de entrenamiento nos hace decantarnos por mantener este parámetro a 16.

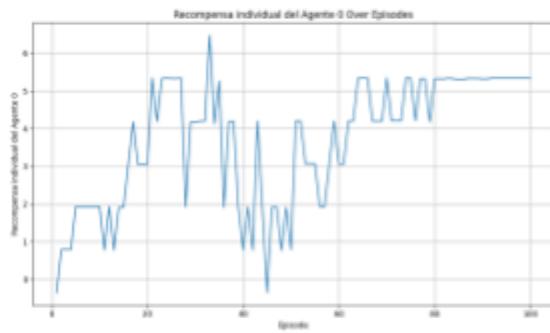
Hipótesis 5: La manipulación del valor alpha no contribuirá a mejorar el rendimiento de nuestros algoritmos. Esta hipótesis se debe a la estabilidad en torno a buenas recompensas obtenidas por nuestra referencia. Dicha estabilidad nos hace pensar que aumentar la tasa de aprendizaje no ayudará a hallar mejores soluciones sino que, más bien, introducirá ruido. Probaremos a incrementarlo de 0.01 a 0.1 de inicio y, en caso de que nos equivoquemos en nuestra hipótesis, seguiremos incrementando mientras optimice los resultados.

Resultados con alpha = 0.1:

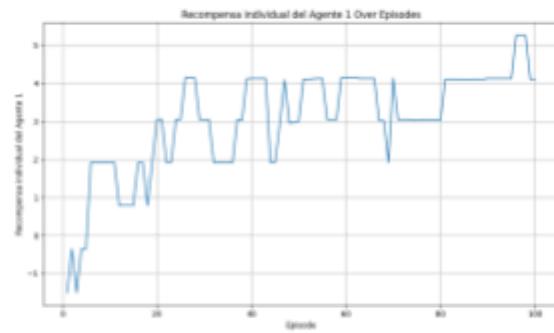
Tiempo de entrenamiento: 0:36



a) Recompensa colectiva



b) Recompensa individual del Agente 0



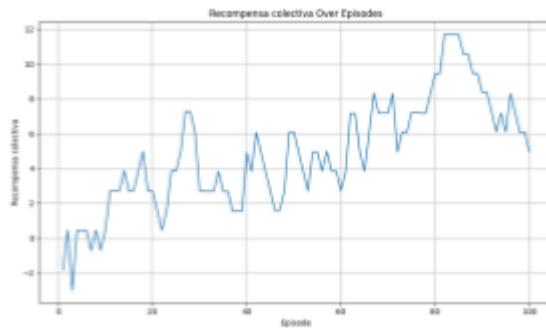
c) Recompensa individual del Agente 1

Tal y como habíamos previsto, el cambio no mejora nuestros algoritmos. Las gráficas resultantes muestran una mayor variabilidad de las recompensas que, en ningún caso, llegan a los altos niveles de recompensas estables que obtenemos con alpha = 0.01. No seguiremos, por ello, con esta exploración. Mantendremos el valor que veníamos usando.

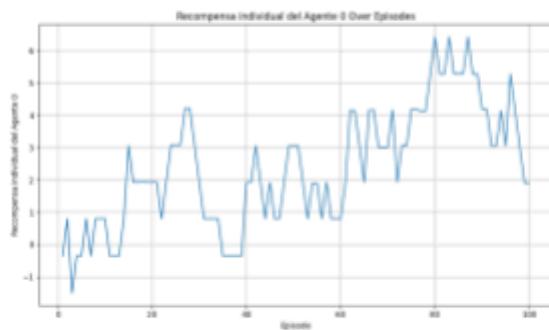
Hipótesis 6: En lo referente al parámetro epsilon sucede igual que en el caso anterior. Los motivos son también los mismos pero en referencia a la posibilidad de explorar nuevas opciones. Iremos incrementando gradualmente epsilon_min en caso de que nos equivoquemos.

Resultados con epsilon_min = 0.3:

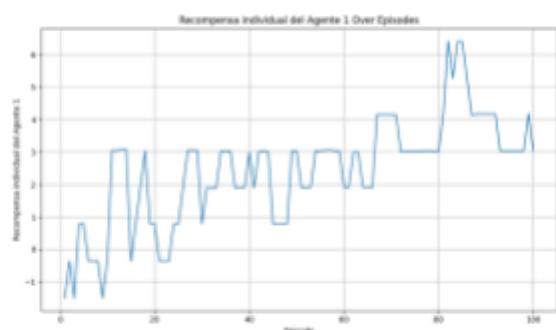
Tiempo de entrenamiento = 0.36



a) Recompensa colectiva



b) Recompensa individual del Agente 0



c) Recompensa individual del Agente 1

Como en el caso anterior, la modificación del parámetro en cuestión no ha contribuido a una mejora en el rendimiento de nuestros algoritmos. Zanjamos, así, la duda sobre la posibilidad de hallar mejores resultados con estos cambios.

Conclusiones:

Tras, progresivamente, modificar los parámetros de nuestro algoritmo en busca de la mejor combinación de estos para resolver nuestro problema, hemos llegado a la conclusión de que la combinación que mejor lo soluciona es:

- epoch = 100
- episodes_per_epoch = 10
- episode_length = 16
- learning_rate = 0.01
- epsilon_max = 1
- epsilon_min = 0.1

Esta lista solo presenta un cambio respecto de los valores originales: hemos reducido el número de epochs a la mitad. Si no hemos hecho más cambios ha sido por no hallar ninguna otra vía combinatoria que nos pareciese satisfactoria para resolver nuestro problema.

2.-¿Cómo se comportan los agentes cuando están entrenados para optimizar su comportamiento respecto de cada concepto de solución incluido (minimax, Nash, wellness, Pareto)? ¿Para qué tipo de coordinación es más adecuado cada uno (intereses comunes, conflictivos, mixtos) ?

Vamos a comparar el comportamiento de los agentes en distintos escenarios en los que utilizaremos distintos conceptos de solución:

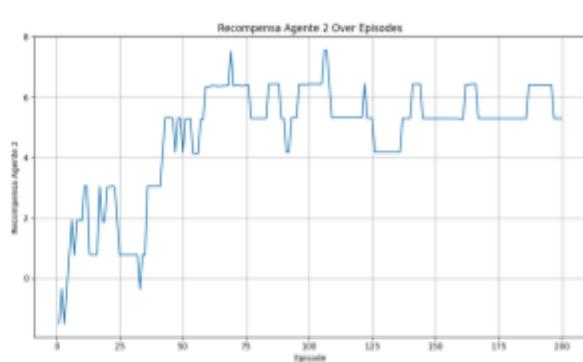
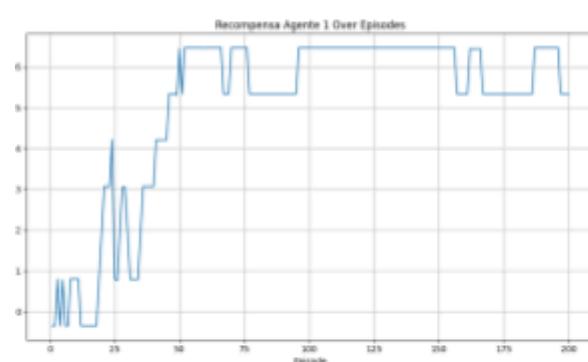
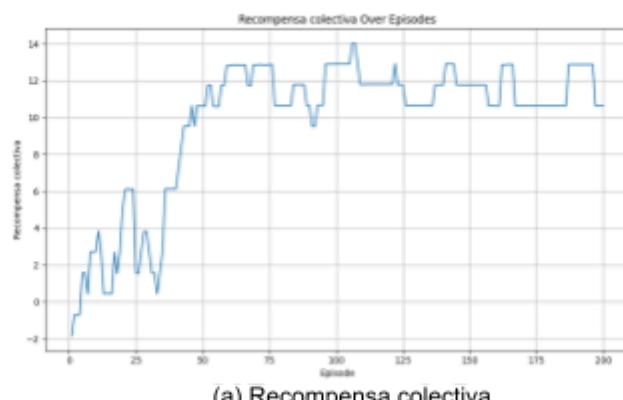
- Minimax
- equilibrio de Nash
- Wellness
- Pareto

Óptimo de Pareto

Vamos a empezar con agentes con concepto de solución óptima de Pareto. Una solución es óptima de Pareto si no es posible mejorar la situación de un agente sin empeorar la de otro, se busca mejorar nuestra situación sin perjudicar a los demás.

Por lo tanto, este concepto de solución se centra en intereses comunes (evita perjudicar al resto) y en intereses mixtos, puesto que busca el equilibrio entre mejorar su situación sin perjudicar a otro.

Hipótesis: En este contexto, se trata de un juego en el que nos interesa maximizar el bien común e individual de cada agente, por lo que creemos que tener agentes con óptimo de Pareto puede llevar a una buena coordinación para obtener buenas recompensas colectivas e individuales.

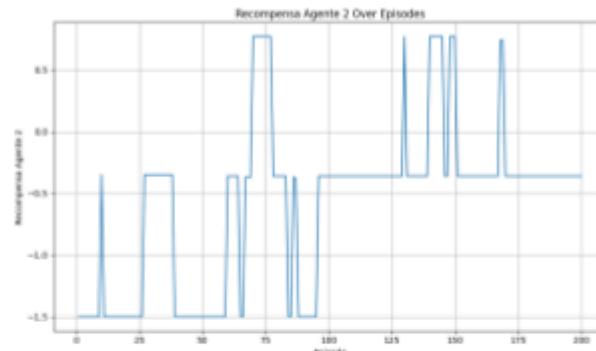
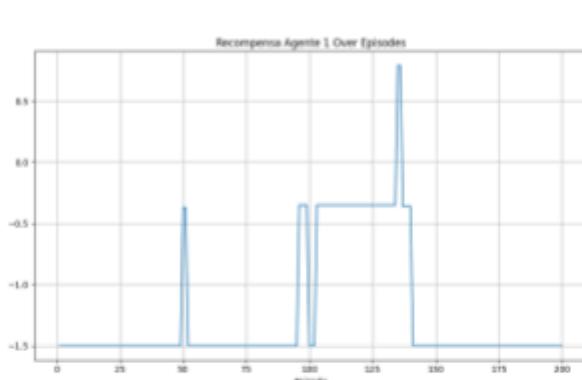
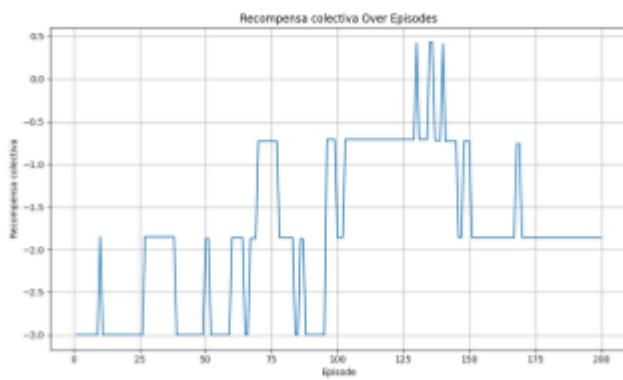


Como podemos observar, ambos agentes cooperan y obtienen recompensas individuales similares altas para una recompensa colectiva elevada. Se confirma nuestra hipótesis, que al tratarse de un problema en el que se busca el interés común y mixto, el óptimo de Pareto es un buen concepto de solución para este fin.

Minimax

Minimax es una estrategia comúnmente utilizada en juegos de suma cero. Es adecuado para juegos de observabilidad total y para entornos donde los agentes tienen intereses completamente opuestos. Minimax no busca mejorar la recompensa colectiva y por lo tanto no es adecuado para entornos cooperativos donde los agentes comparten objetivos.

Hipótesis: El entorno de POGEMA no es totalmente observable, es parcialmente observable, y se busca maximizar la recompensa colectiva, por lo que los objetivos de Minimax son totalmente opuestos a nuestro problema. Por ende, creemos que utilizar Minimax nos dará recompensas muy bajas tanto a nivel colectivo como individual.



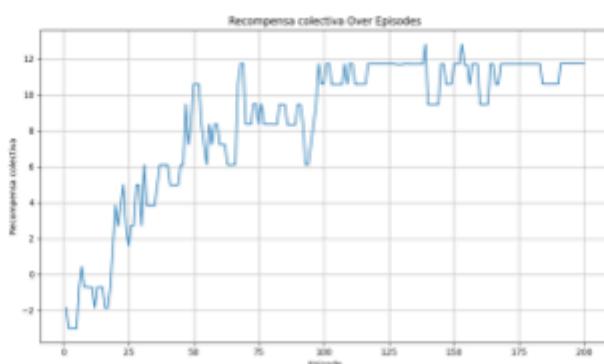
Las recompensas del agente 1 son mayoritariamente negativas, de igual manera que las del agente que no llegan al valor de uno. Por consiguiente, las recompensas colectivas, una vez han convergido, oscilan entre -2 y 0.5, valores muy inferiores a los resultados encontrados en el anterior experimento con el óptimo de Pareto. Podemos entonces concluir que, tal y como esperábamos, la estrategia Minimax no es para nada adecuada con nuestro problema.

Equilibrio de Nash

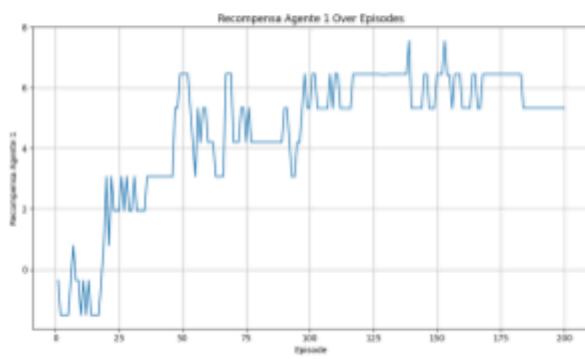
En el Equilibrio de Nash, cada agente selecciona su mejor estrategia, sin tener incentivo para desviarse unilateralmente, siempre y cuando el resto de agentes no cambie de estrategia. Tienden a buscar estrategias estables donde ninguna parte tiene incentivo de cambiar de acción de manera unilateral.

Es adecuado para entornos con intereses mixtos o conflictivos.

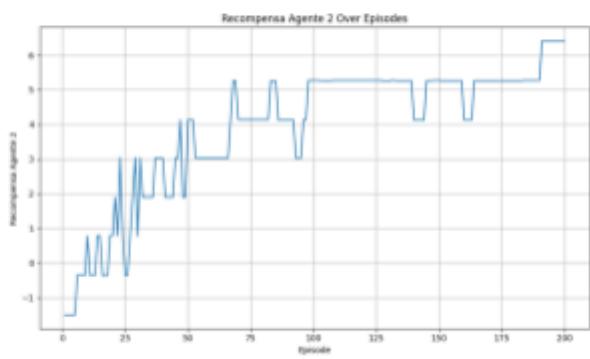
Hipótesis: Nuestro entorno tiene intereses mixtos, tal y como busca el equilibrio de Nash, pero no tiene intereses puramente conflictivos. Por lo tanto, creemos que el equilibrio de Nash puede ser una buena opción para el entorno de POGEMA, que contiene intereses mixtos, pero quizás no es tan buena opción como el óptimo de Pareto, puesto que nuestro entorno en ningún caso tiene intereses conflictivos.



(a) Recompensa colectiva



(b) Recompensa agente 1 (Nash)



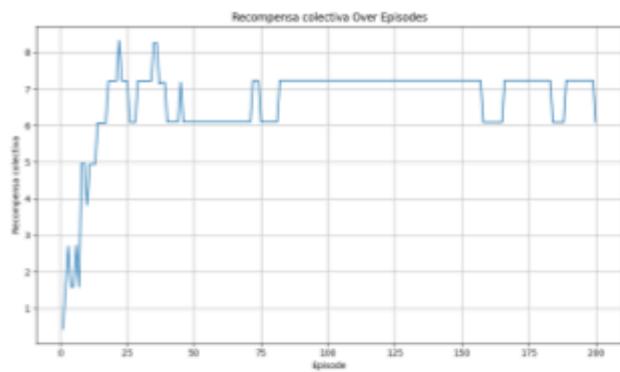
(c) Recompensa agente 2 (Nash)

Como podemos observar, las recompensas colectivas convergen a valores que oscilan entre 10 y 12, que son valores altos, aunque ligeramente inferiores a las obtenidas con el óptimo de Pareto, tal y como suponíamos. Parece ser que los 2 agentes colaboran efectivamente, obteniendo recompensas altas similares, aunque siendo precisos las del agente 1 son ligeramente superiores. Pero ambos agentes colaboran por el bien común sin buscar el detrimento del otro.

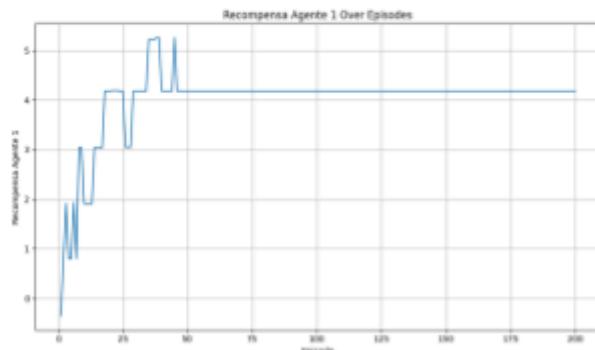
Wellness

El objetivo de wellness es maximizar el bienestar colectivo, que viene definido por la suma de las recompensas de todos los agentes. Los agentes colaboran para buscar el interés común en lugar del individual. Es ideal para entornos totalmente cooperativos donde los agentes buscan maximizar puramente la recompensa colectiva.

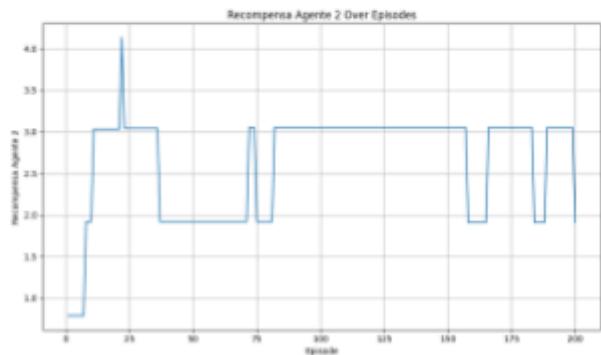
Hipótesis: Dado que nuestro entorno busca maximizar las recompensas comunes, el concepto de solución wellness podría dar recompensas comunes altas. No obstante, los agentes en ningún caso buscan su beneficio individual (que en nuestro caso contribuye al beneficio común), por lo que es posible que las recompensas de los agentes no sean similares y haya cierto desnivel entre sus recompensas individuales.



(a) Recompensa colectiva



(b) Recompensa agente 1 (Wellness)



(c) Recompensa agente 2 (Wellness)

Las recompensas colectivas convergen hacia valores que oscilan entre 6 y un poco más de 7, lo que son valores altos comparados con el inadecuado concepto de solución Minimax visto anteriormente. Pero estas recompensas son menores que las que hemos encontrado con el óptimo de Pareto y el equilibrio de Nash. Esto podría suceder porque *wellness* solo se centra exclusivamente en intereses comunes, pero en nuestro entorno también son necesarias las recompensas individuales.

Además, como vemos, tal y como hemos comentado en la hipótesis, las recompensas del agente 1 superan el valor 4, pero las del agente 2 oscilan entre 2 y 3. Esto no es ilógico porque *wellness* solo se centra en las recompensas comunes, lo que puede provocar esta diferencia en las recompensas individuales.

Conclusiones

En definitiva, hemos podido comprobar que como Minimax es adecuado para intereses conflictivos, no ha sido adecuado para nuestro entorno y nos ha dado, con diferencia, los peores resultados. *Wellness*, que sólo se centra en intereses comunes, nos ha dado resultados colectivos aceptables, pero que han sido inferiores a los que nos ha dado el equilibrio de Nash, el cual se centra en intereses mixtos y conflictivos.

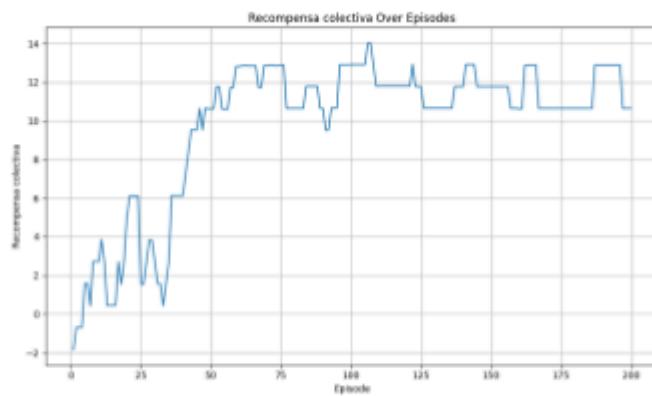
Finalmente, el ganador en cuanto a recompensas comunes es el concepto de solución del óptimo de Pareto, que se centra en intereses mixtos y comunes.

3.- ¿Qué ocurre si entrenamos a dos agentes con dos conceptos de solución diferentes? ¿Son capaces de coordinarse, o de conseguir recompensas individuales independientemente de la política del otro agente?

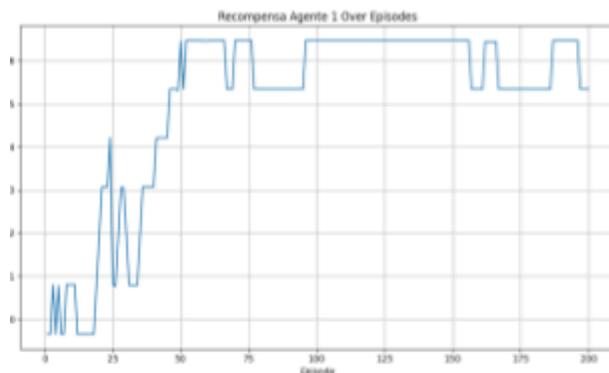
Para este experimento utilizaremos 2 agentes. A uno de ellos le vamos a asignar el concepto de solución del óptimo de Pareto, y al otro el equilibrio de Nash.

Hipótesis: El agente Pareto busca mejorar SIN que empeore ninguno de los otros agentes, en cambio el agente Nash busca maximizar su recompensa si los otros agentes no cambian de estrategia. Por un lado, el óptimo de Pareto podría buscar más el beneficio común, mientras que quizás Nash podría centrarse más en su propio beneficio, llevando a una falta de cooperación entre ambos agentes que lleve al sistema multiagente a una recompensa común menor.

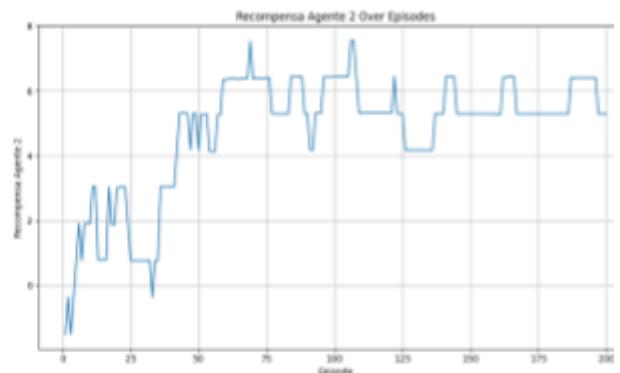
En primer lugar, como punto de referencia escogemos 2 agentes que utilizarán el óptimo de Pareto.



(a) Recompensa colectiva

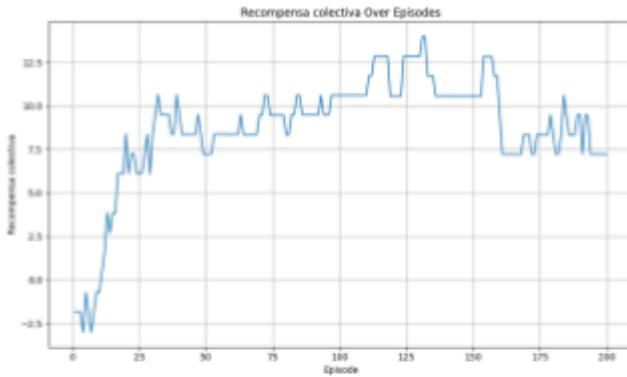


(b) Recompensa agente 1 (Pareto)

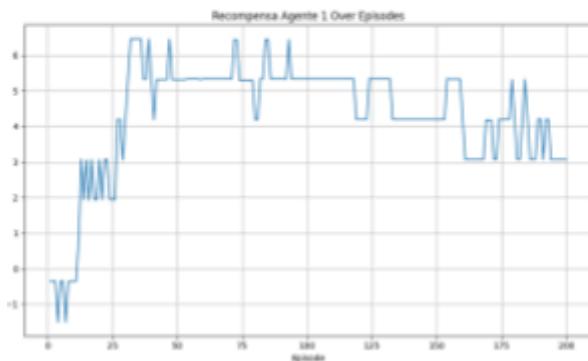


(c) Recompensa agente 2 (Pareto)

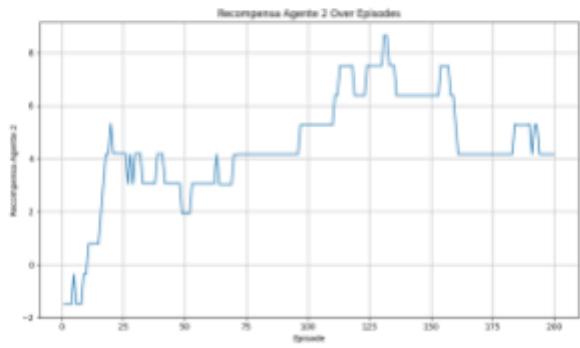
Ahora, al agente 1 le asignamos el óptimo de Pareto y al agente 2 el equilibrio de Nash.



(a) Recompensa colectiva



(b) Recompensa agente 1 (Pareto)

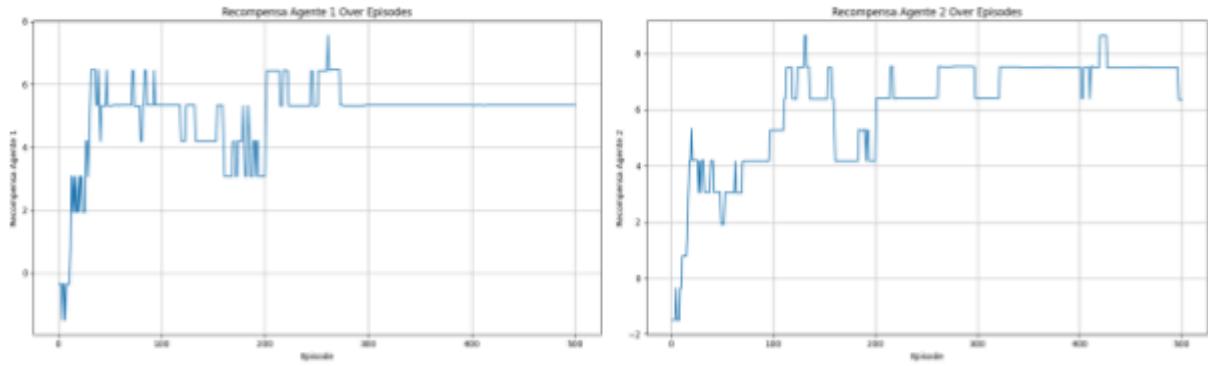


(c) Recompensa agente 2 (Nash)

Como podemos observar, cuando los 2 agentes utilizan el mismo concepto de solución (en este caso: Pareto), la recompensa común oscila entre mayoritariamente entre 11 y 13. En cambio, cuando los 2 agentes utilizan conceptos de solución distintos, su falta de cooperación nos lleva a una recompensa común menor de entre 7,5 y 12,5, tal y como suponíamos en la hipótesis.

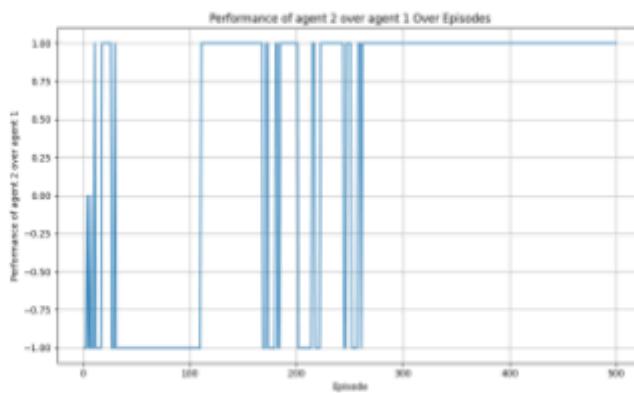
Ahora, centrémonos en las recompensas individuales del agente Pareto y las del agente Nash. Para ver si se confirma nuestra hipótesis en la que el agente Nash podría obtener recompensas mayores que Pareto, observamos las gráficas (b) y (c) de la figura superior. No podemos concluir con certeza que el agente Nash está obteniendo mayores recompensas que Pareto, puesto que hay bastante divergencia. Podría ser que el número de iteraciones no sea suficiente y no se haya convergido suficientemente. Así que vamos a hacer un último intento en el que aumentamos el número de iteraciones de 200 a 500, y generamos un gráfico con valores que significan lo siguiente:

- 1 → el agente Nash obtiene recompensa mayor que Pareto (lo que queremos)
- 0 → ambos agentes obtienen la misma recompensa
- -1 → el agente Pareto obtiene una recompensa mayor Nash



(a) Recompensa agente 1 (Pareto)

(b) Recompensa agente 2 (Nash)



(c) Recompensa agente 2 sobre agente 1:

$1 \rightarrow$ recompensa Nash > recompensa Pareto; $0 \rightarrow$ mismas recompensas; $-1 \rightarrow$ recompensa Pareto > recompensa Nash

Ahora que hemos ampliado el número de iteraciones, ya se puede observar que el sistema acaba convergiendo a un punto en el que el agente Nash obtiene recompensas ligeramente mayores a las del agente Pareto.

4.- ¿Cómo se compara JAL-GT, que es un algoritmo multiagente debido a que calcula la Q en base a acciones conjuntas, con IQL (independent QLearning)?

Primero haremos una comparación general:

JAL-GT es multiagente, lo que significa que considera las acciones de todos los agentes al calcular la función Q. Esto le permite aprender estrategias de coordinación y cooperación entre los agentes. JAL-GT utiliza conceptos de la teoría de juegos (como minimax, equilibrio de Nash, etc.) para decidir la mejor acción conjunta en cada estado.

Por otro lado, IQL es independiente para cada agente, lo que significa que cada agente aprende su propia función Q sin considerar las acciones de los demás. Cada agente actúa de manera egoísta, buscando maximizar su propia recompensa sin tener en cuenta el impacto en los otros agentes.

Hablando de la recompensa colectiva, se espera que JAL-GT obtenga una recompensa colectiva más alta que IQL, ya que JAL-GT puede aprender estrategias de coordinación que beneficien a todos los agentes. IQL, por otro lado, puede llevar a que los agentes se bloquen entre sí o tomen decisiones que no sean óptimas para el grupo. Del mismo modo, al hablar de recompensa individual, la recompensa individual de los agentes en JAL-GT puede ser menor que en IQL en algunos casos. Esto se debe a que JAL-GT puede sacrificar la recompensa individual de algunos agentes en favor de una mayor recompensa colectiva. IQL, al ser egoísta, puede permitir que algunos agentes obtengan mayores recompensas individuales a expensas de otros.

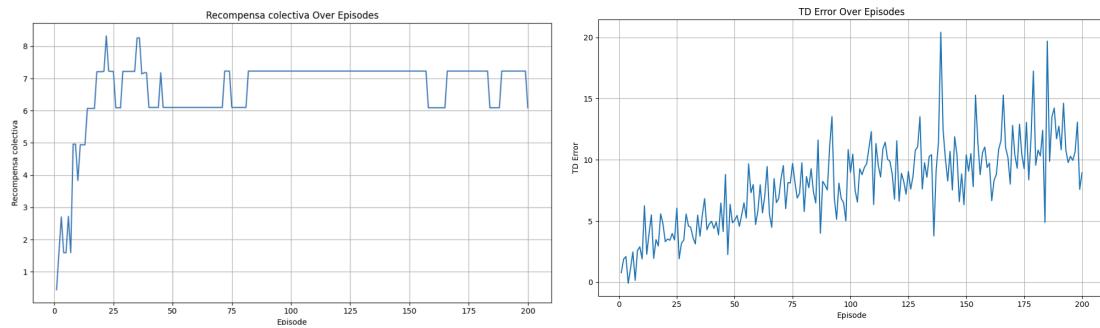
Por último, a nivel de tiempo, JAL-GT puede requerir más tiempo de entrenamiento que IQL debido a la mayor complejidad de calcular la función Q para acciones conjuntas. IQL, al ser independiente para cada agente, puede converger más rápidamente.

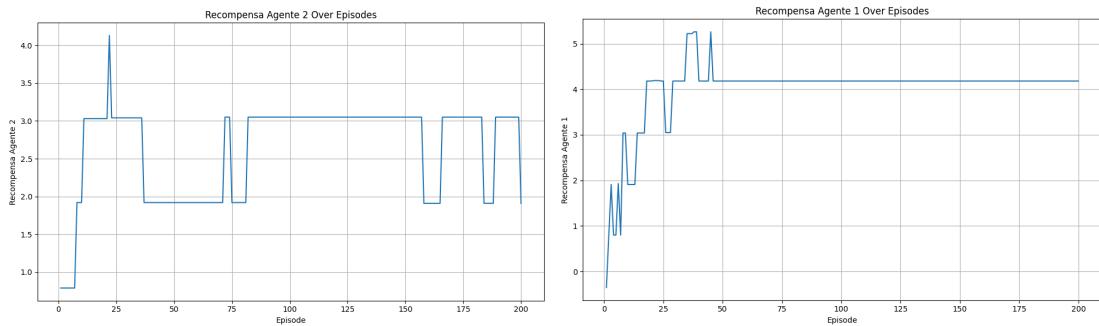
Procedemos a hacer la comparación entre los algoritmos JAL-GT y IQL (Independent Q-Learning), viendo las gráficas de los diferentes algoritmos para distintos conceptos de solución, con 2 agentes. Una vez hecho esto, procederemos a hacer lo mismo para 4 agentes, para comprobar los puntos que mencionaremos:

Resultados con 2 agentes:

Ejecución con JALGT y WelfareSolutionConcept.

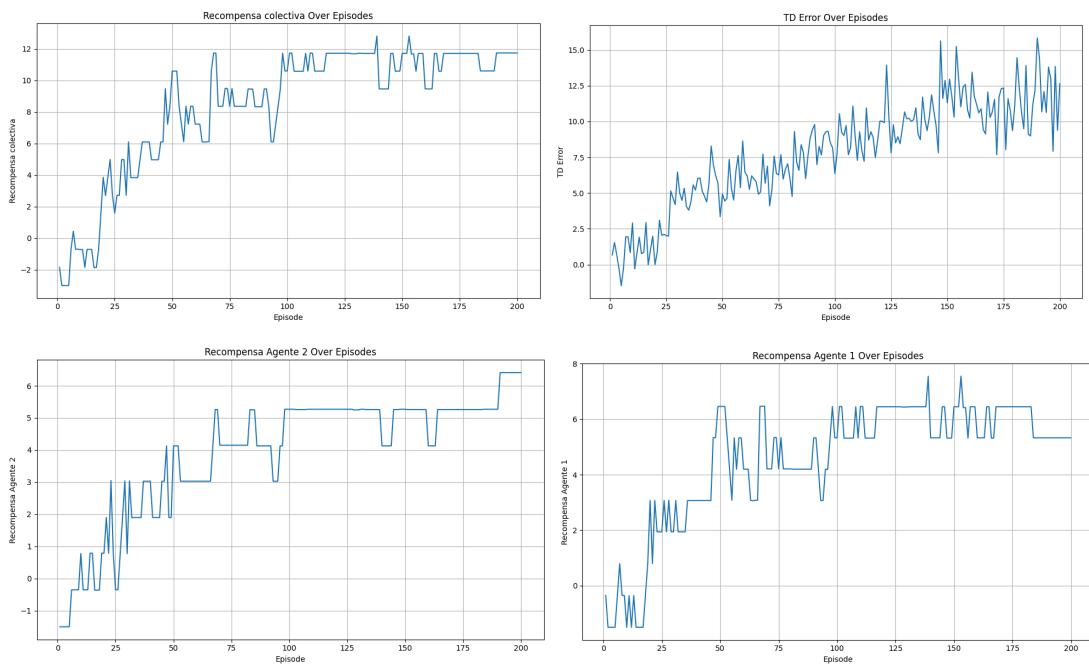
Recompensa colectiva del último epoch de 6.09 y tiempo : 160 segundos:





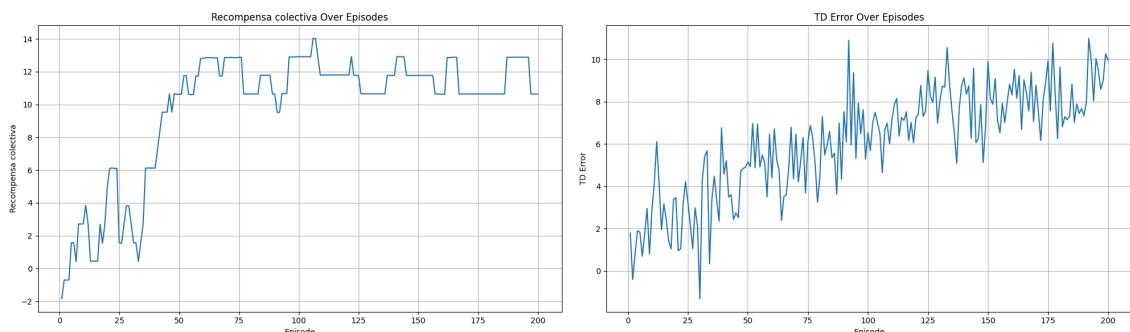
Ejecución con JALGT y NashSolutionConcept. 2 agentes.

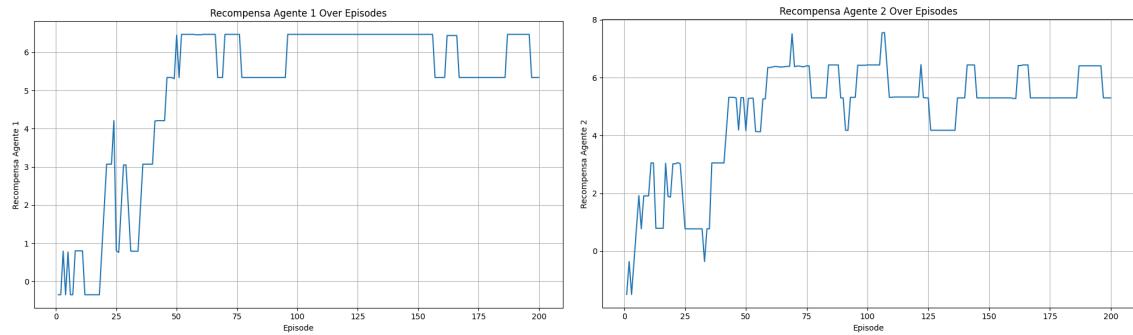
Recompensa colectiva del último epoch de 11.74 y tiempo : 144 segundos:



Ejecución con JALGT y ParetoSolutionConcept. 2 agentes.

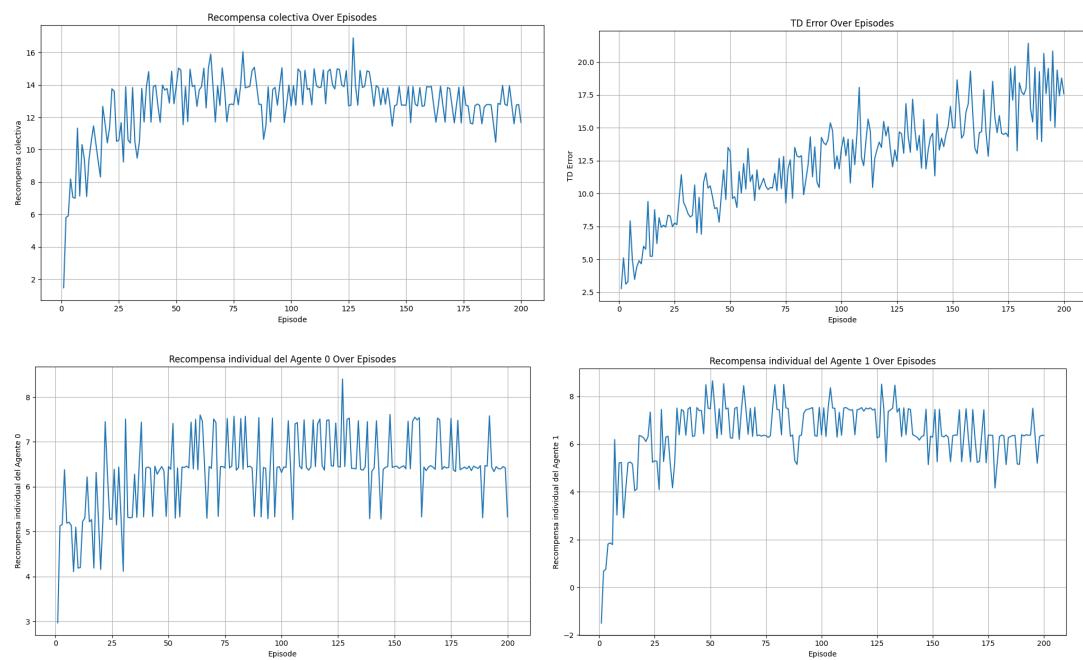
Recompensa colectiva del último epoch de 10.64 y tiempo : 210 segundos:





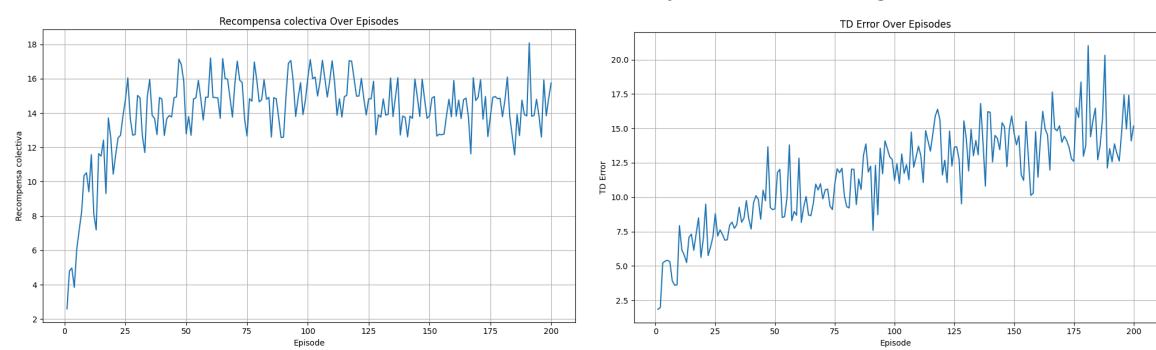
Ejecución con IQL y ParetoSolutionConcept. 2 agentes.

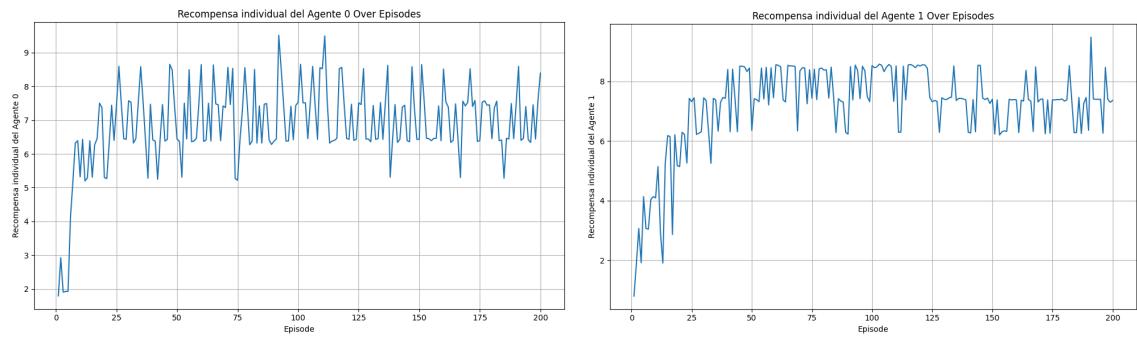
Recompensa colectiva del último epoch de 11.690 y tiempo : 80 segundos:



Ejecución con IQL y NashSolutionConcept. 2 agentes.

Recompensa colectiva del último epoch de 15.750 y tiempo : 82 segundos:





Conclusiones con 2 agentes:

Refiriéndonos a los puntos mencionados anteriormente, en la introducción general a la comparación, el punto dónde más se puede confirmar es en el tiempo, que según los resultados vemos claramente que JAL-GT tarda más. En el sentido de recompensas individuales y colectivas, al ser sólo dos agentes, no se puede apreciar una diferencia significativa y por lo tanto procedemos a aumentar el número de agentes a 4.

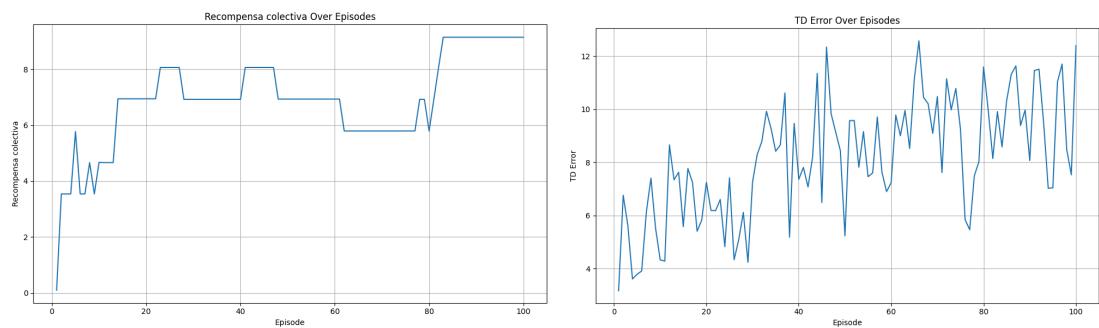
Resultados con 3 y 4 agentes:

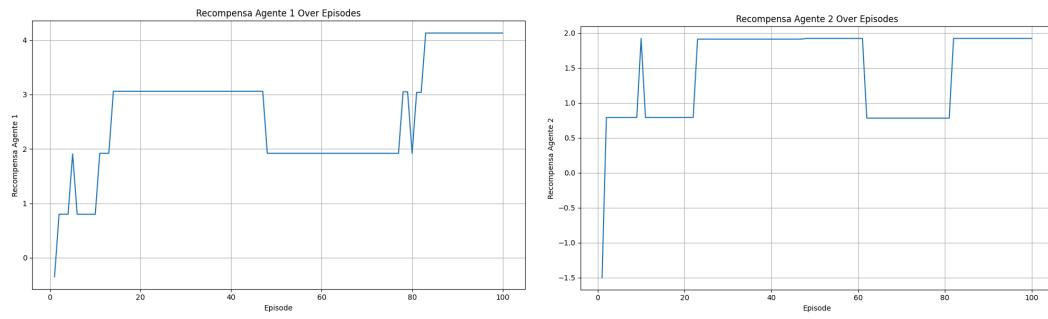
Para este apartado de la comparación, hemos decidido ejecutar JAL-GT con número de epochs 100 y IQL con 200, y JAL-GT con 3 agentes mientras que IQL lo ejecutamos con 4. Esto es porque JAL-GT con los parámetros de IQL tarda un tiempo demasiado largo, y los resultados que nos interesan del experimento es ver cómo escalan las afirmaciones generales sobre los algoritmos en la introducción de esta pregunta:

El hecho de no poder incrementar el número de agentes se ve reforzado en la siguiente pregunta.

Ejecución con JAL-GT y ParetoSolutionConcept. 3 agentes.

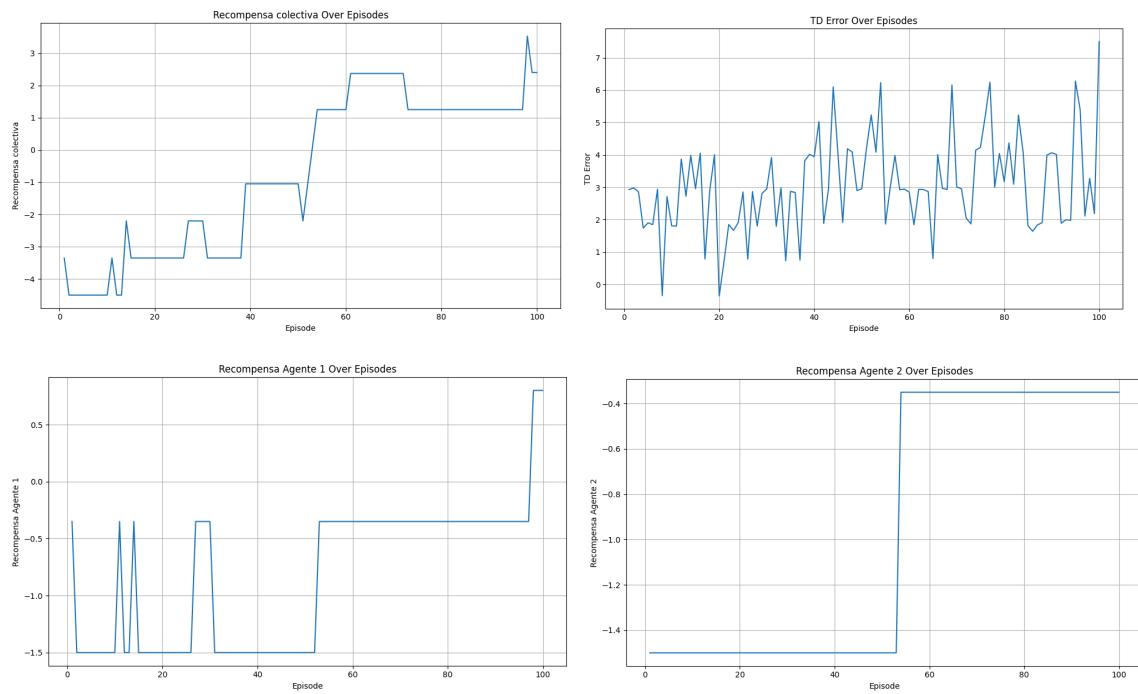
Recompensa colectiva del último epoch de 9.13 y tiempo : 360 segundos:





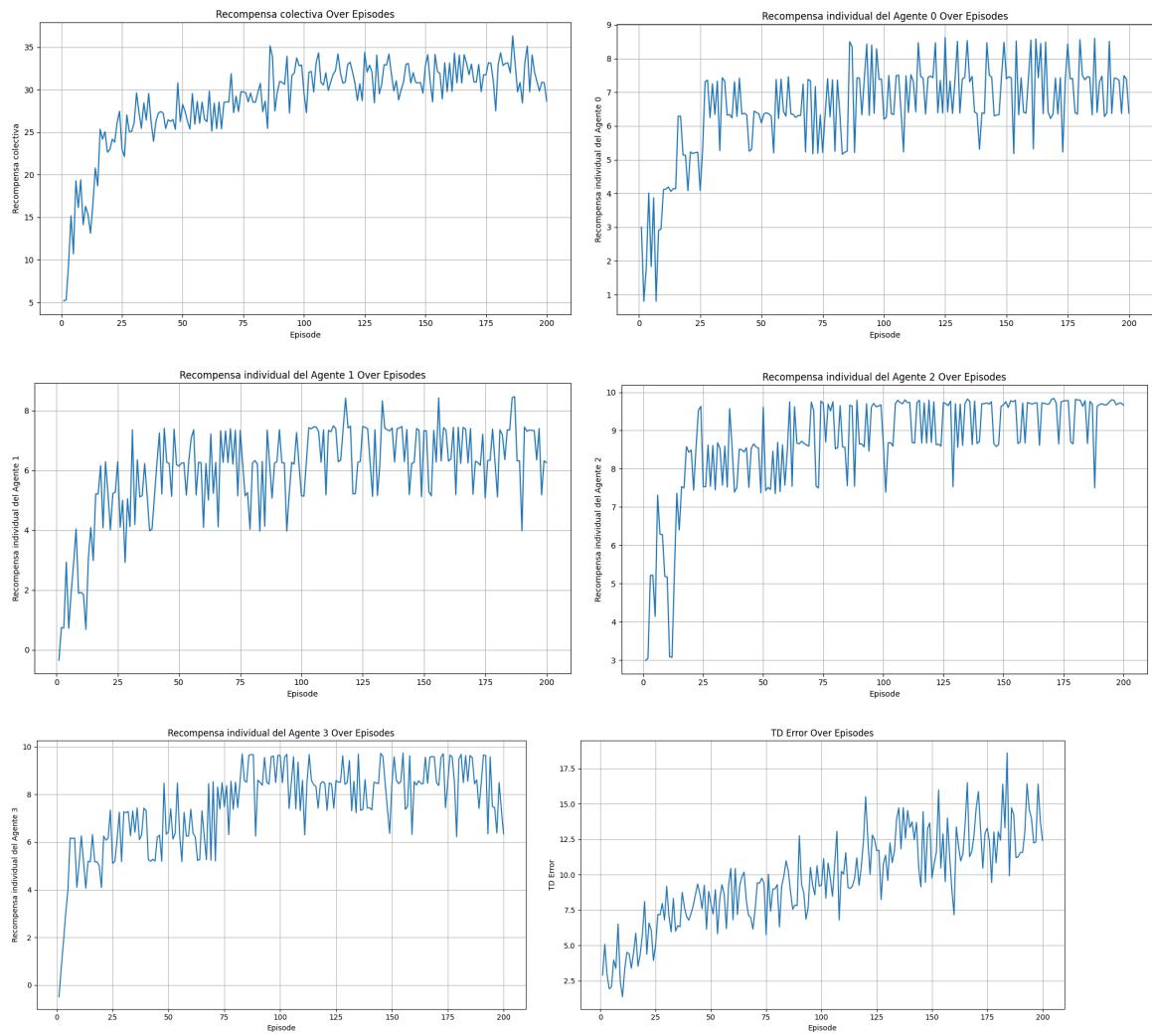
Ejecución con JAL-GT y NashSolutionConcept. 3 agentes.

Recompensa colectiva del último epoch de 2.400 y tiempo : 360 segundos:



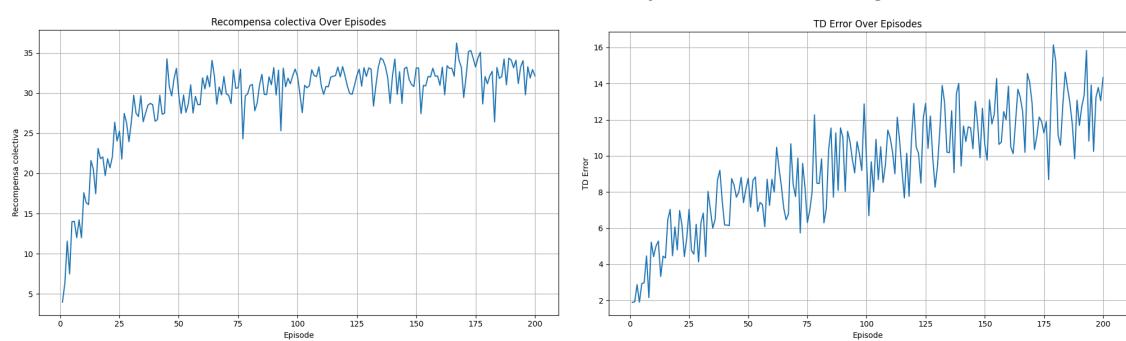
Ejecución con IQL y ParetoSolutionConcept. 4 agentes.

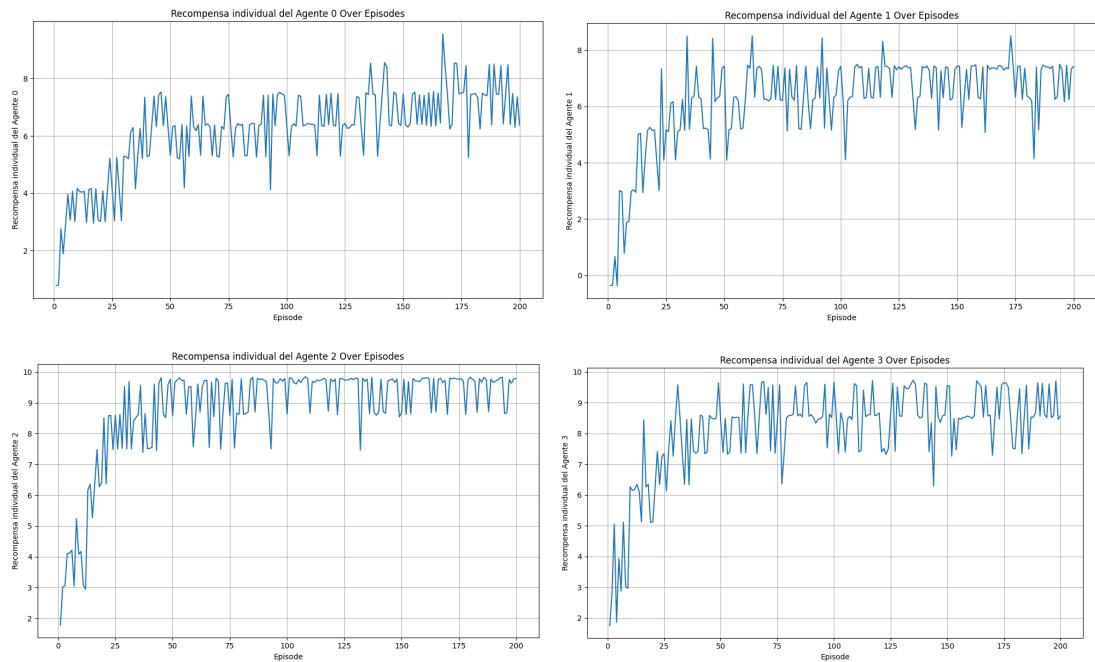
Recompensa colectiva del último epoch de 28.64 y tiempo : 186 segundos:



Ejecución con IQL y NashSolutionConcept. 4 agentes.

Recompensa colectiva del último epoch de 32.15 y tiempo : 172 segundos:





Conclusiones con 3 y 4 agentes:

Para acabar con esta pregunta, podemos comprobar mediante las gráficas, de manera más clara, por ejemplo en estas últimas gráficas, que al actuar de manera egoísta, se consiguen peores recompensas colectivas, y a nivel individual un agente despuña ante el resto, debido a que no cooperan. Por otra parte, el tiempo sigue siendo claramente superior en JAL-GT.

Lo que sí que podemos ver en JAL-GT es el hecho que se potencia la recompensa colectiva al máximo posible.

En resumen, aunque a priori y a nivel teórico, JAL-GT debería dar mejores resultados ya que los agentes son capaces de cooperar entre sí, en este entorno con estas condiciones concretas, aunque podemos ver alguna pincelada de las características de los algoritmos, funciona claramente mejor IQL, la cuál cosa se ve reflejada en las recompensas obtenidas.

5.-¿Cómo es capaz de generalizar el algoritmo? Es decir: con el resto de parámetros fijados, ¿cómo es capaz el algoritmo de converger a medida que escala el tamaño del entorno (2×2, 3×3, 4×4, 8×8...), el número de agentes (2, 3, 4...), la complejidad de la representación del estado, etc.?

Para responder a esta cuestión partiremos como base de los resultados obtenidos en la pregunta 1. Los parámetros referentes al modelo son los especificados en las conclusiones de dicha pregunta. Aquí nos ocuparemos de ver cómo escala la complejidad del entrenamiento al aumentar el tamaño del entorno, el número de agentes y la densidad de obstáculos.

Escalado del número de agentes:

Resultados con 3 agentes:

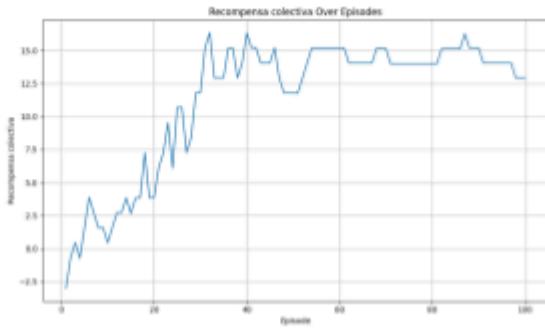
Al ampliar el problema, a partir de los parámetros iniciales, a 3 agentes, se tardan 113.36 segundos en completar una iteración de entrenamiento en nuestra máquina, por lo que completar un entrenamiento de 100 iteraciones se prevé que tarde en torno a las 3 horas y 7 minutos. Esto no permite escalar el problema a título experimental en esta dimensión. Sencillamente, la complejidad crece demasiado. No podemos seguir explorando este camino.

Escalado del tamaño del entorno:

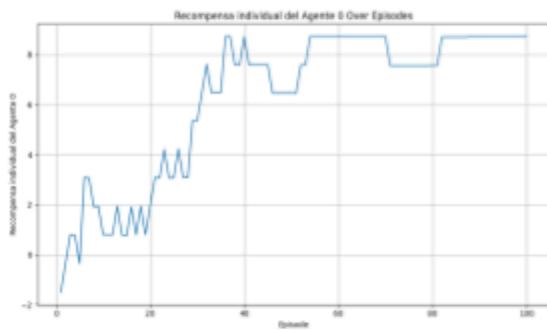
El entorno original tenía por parámetro size = 4

Resultados con size = 3:

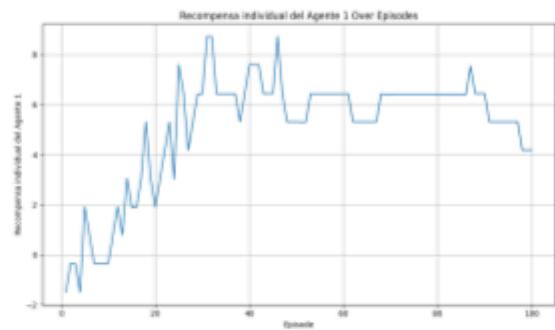
Tiempo de entrenamiento: 0:31



a) Recompensa colectiva



b) Recompensa individual del Agente 0

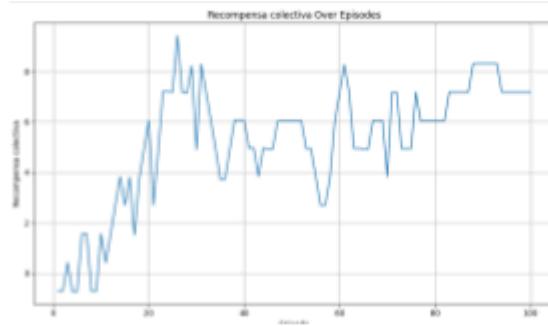


c) Recompensa individual del Agente 1

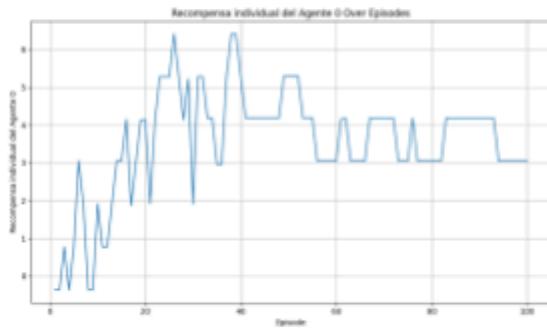
En este caso, nuestros algoritmos rinden tan bien como antes e, incluso, obtienen mejores recompensas dada la mayor facilidad asociada a resolver este problema. El tiempo de ejecución se mantiene en el mismo orden de magnitud.

Resultados con size = 6:

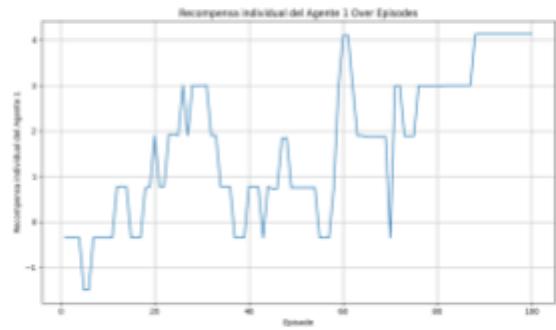
Tiempo de entrenamiento: 0:46



a) Recompensa colectiva



b) Recompensa individual del Agente 0

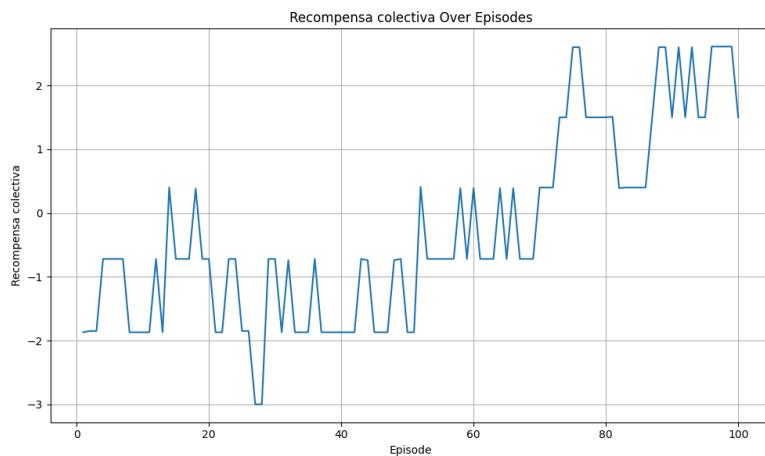


c) Recompensa individual del Agente 1

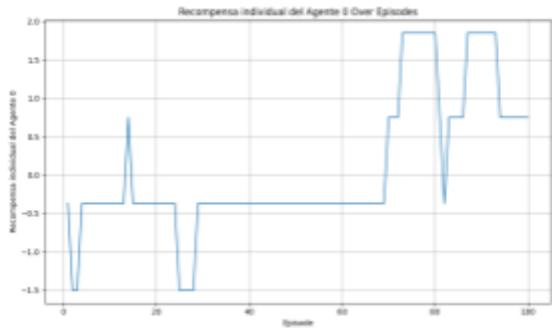
El algoritmo sigue rindiendo adecuadamente pese a un notable decremento en las recompensas debido a la mayor dificultad de resolución que un entorno mayor implica.

Resultados con size = 8:

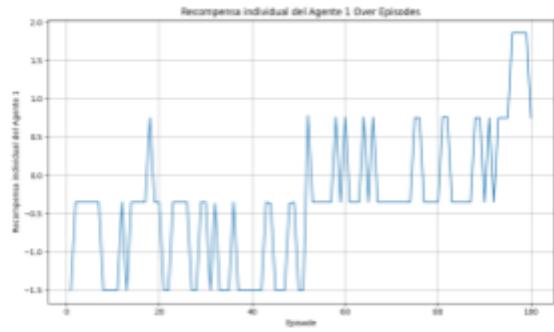
Tiempo de entrenamiento: 0:44



a) Recompensa colectiva



b) Recompensa individual del Agente 0



c) Recompensa individual del Agente 1

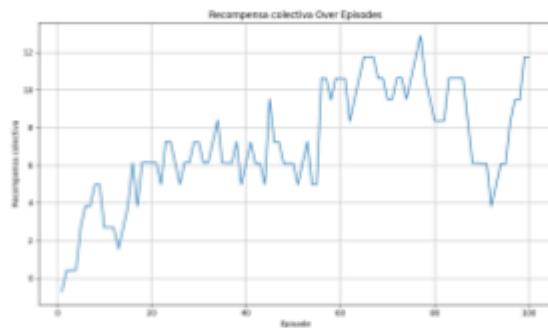
Al ampliar el tamaño del entorno a 8, nuestra parametrización del algoritmo ya no es capaz de resolver el problema que nos ocupa. Si bien el tiempo de entrenamiento es bueno, no basta para resolver nuestro problema. Dado que nos habíamos propuesto no modificar más que un parámetro a la vez, concluimos aquí la exploración en esta dimensión afirmando que, si bien es computable, el problema se vuelve muy complejo como para que nuestros algoritmos lo resuelvan.

Escalado de densidad de obstáculos:

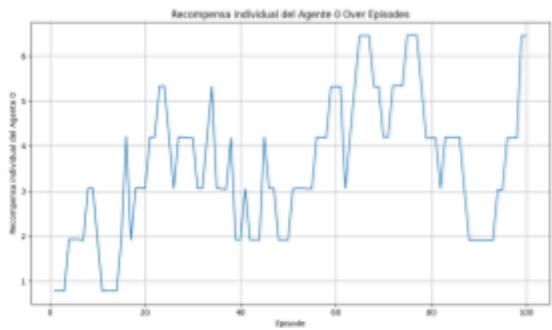
Partíamos de una densidad de 0.1, iremos aumentándola progresivamente.

Resultados con obstacle_density = 0.2

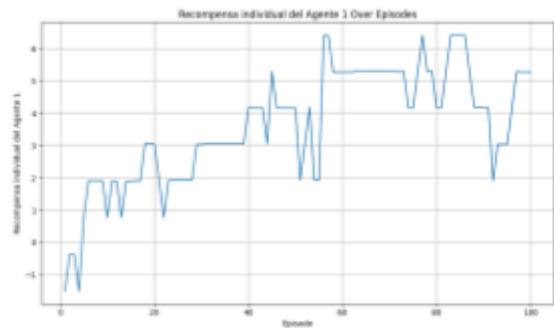
Tiempo de entrenamiento: 0:36



a) Recompensa colectiva



b) Recompensa individual del Agente 0

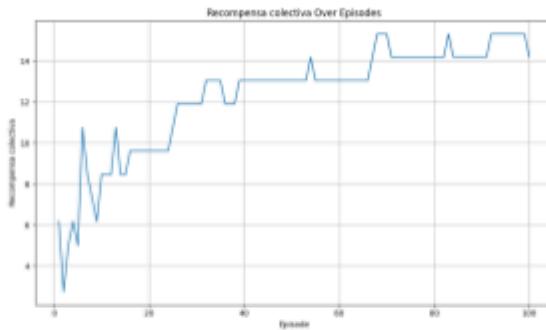


c) Recompensa individual del Agente 1

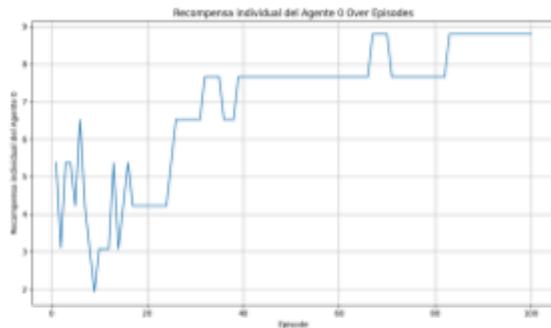
Se puede observar en las gráficas que nuestro algoritmo sigue rindiendo satisfactoriamente pese al incremento en la dificultad del problema. El tiempo de entrenamiento no escala desmesuradamente.

Resultados con obstacle_density = 0.4

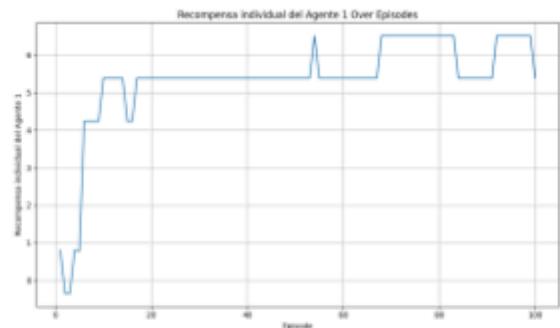
Tiempo de entrenamiento: 0:36



a) Recompensa colectiva



b) Recompensa individual del Agente 0

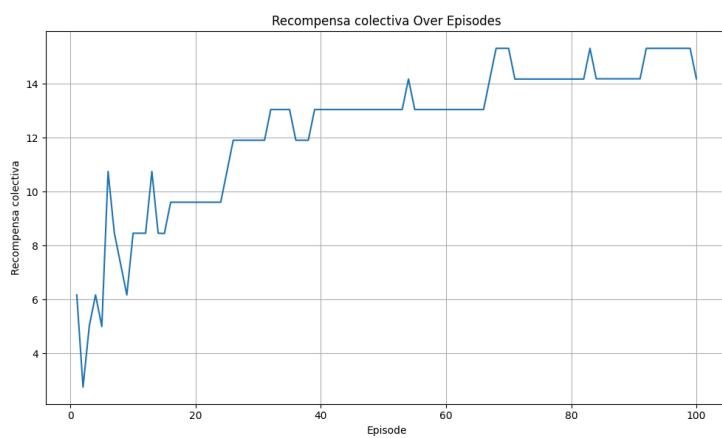


c) Recompensa individual del Agente 1

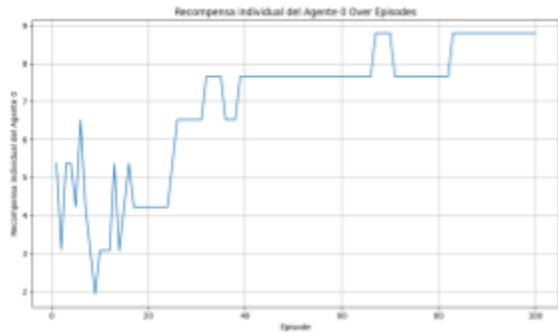
Para nuestra sorpresa, al aumentar la densidad de obstáculos en el entorno nuestros agentes resuelven el problema todavía mejor de lo que lo hacían, obteniendo mayores recompensas colectivas e individuales sin por ello aumentar el tiempo de entrenamiento.

Resultados con obstacle_density = 0.8

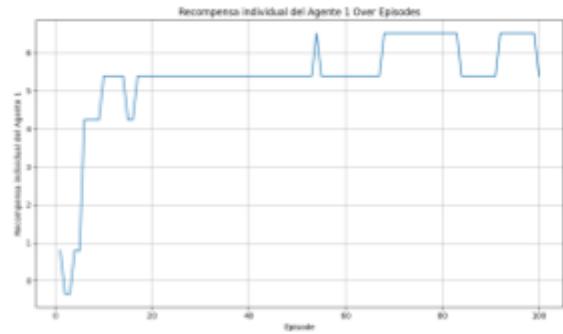
Tiempo de entrenamiento: 0:28



a) Recompensa colectiva



b) Recompensa individual del Agente 0

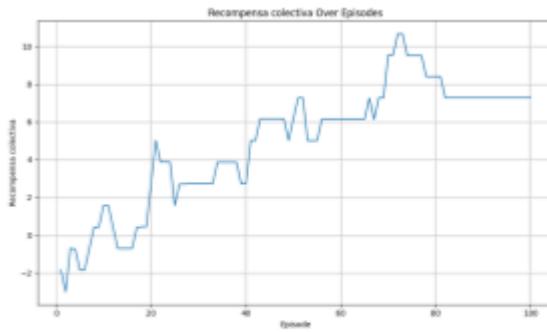


c) Recompensa individual del Agente 1

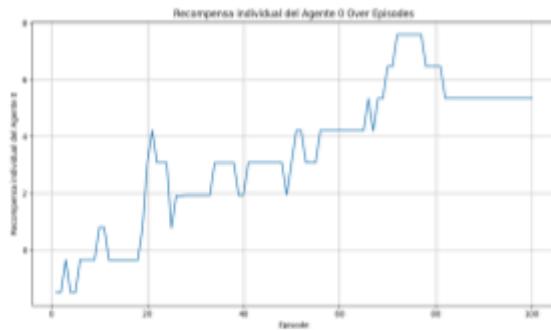
En este último experimento se constata la dinámica acuciada ya en el anterior. Aumentar la densidad de obstáculos facilita, contrariamente a lo que pensábamos, el resolver el problema. Esto se debe, seguramente, al mayor bloqueo de rutas posibles por parte de los obstáculos que reduce, asimismo, las interferencias entre agentes.

6.-¿Es capaz el algoritmo de generalizar de manera que tenga éxito evaluando mapas que no ha visto en entrenamiento?

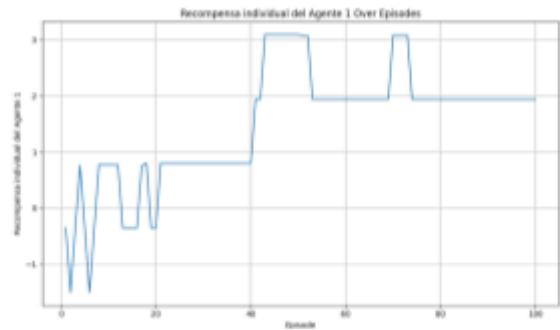
Para efectuar esta comprobación sólo debemos modificar levemente el código de main.py para garantizar que en la evaluación se usan seeds distintas a las del entrenamiento.



a) Recompensa colectiva



b) Recompensa individual del Agente 0



c) Recompensa individual del Agente 1

En líneas generales podemos afirmar que el algoritmo es capaz de generalización y que resuelve adecuadamente el problema aún para mapas que no ha visto en la fase de entrenamiento. Aunque resulta evidente que no rinde tan bien como cuando se evaluaba sobre los mapas con que había entrenado, su rendimiento mejora a medida que avanzan las epochs, llegando a un nivel de rendimiento colectivo adecuado, bueno para el agente 0 y, eso es cierto, no tan óptimo para el Agente 1, que tiene dificultades para completar la tarea.