Trabajo Fin de Grado

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Control de robots móviles mediante aprendizaje por refuerzo

Autor: Beatriz Melara Urbano

Tutor: Federico Cuesta Rojo

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Control de robots móviles mediante aprendizaje por refuerzo

Autor:
Beatriz Melara Urbano

Tutor:
Federico Cuesta Rojo
Profesor titular

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2024

Trabajo Fin de Grado: Control de robots móviles mediante aprendizaje por refuerzo

Autor: Beatriz Melara Urbano

Tutor: Federico Cuesta Rojo

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2024

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que han estado a mi lado durante estos años en la Escuela Técnica Superior de Ingeniería de Sevilla. Este Trabajo de Fin de Grado no habría sido posible sin su apoyo constante y ánimo.

Primero, quiero agradecer a mis padres, que siempre han creído en mí y me han apoyado incondicionalmente en cada paso del camino. Su amor, paciencia y sacrificio han sido la base de todos mis logros académicos. A mi hermano, por su aliento constante y por estar siempre dispuesto a escucharme y darme consejos cuando más los necesitaba.

A Adrián, gracias por tu comprensión, cariño y por ser mi refugio en los momentos más difíciles. Tu apoyo y motivación han sido esenciales para mí.

A mis amigos Carlota y Juan Carlos, cuya amistad ha sido un pilar fundamental durante estos años. Carlota, gracias por ser una amiga leal y por compartir tantas risas y buenos momentos. Juan Carlos, mi gran amigo y compañero de días intensos de biblioteca, gracias por tu compañía y por hacer que las largas jornadas de estudio fueran más llevaderas, e incluso divertidas.

Quiero también agradecer a los profesores de la Escuela Técnica Superior de Ingeniería de Sevilla por su dedicación y por inspirarme a alcanzar mis metas académicas. Su compromiso con la enseñanza y su disposición para ayudarme en todo momento han sido clave para mi formación.

Quiero agradecer al Colegio Mayor Hernando Colón, mi hogar durante estos años de estudio. Allí encontré un ambiente acogedor, así como personas maravillosas que me permitieron crecer tanto académicamente como personalmente.

Por último, un agradecimiento especial a mi tutor, Federico, por su guía y apoyo durante la realización de este trabajo. Su conocimiento, paciencia y consejos han contribuido significativamente a la finalización exitosa de este proyecto.

Beatriz Melara Urbano

Sevilla, 2024

Resumen

En el ámbito de la investigación y aplicación de técnicas de aprendizaje por refuerzo, es común el uso de algoritmos avanzados para el entrenamiento de robots móviles en entornos simulados. Estos algoritmos, como el Q-Learning y Sarsa, son fundamentales en el aprendizaje por refuerzo. Con el desarrollo de este Trabajo de Fin de Grado, se busca implementar y comparar estos dos algoritmos para mejorar el rendimiento de robots en tareas específicas.

Para conseguir esto, se propone, en primer lugar, la configuración del entorno de simulación utilizando herramientas como Gazebo, ROS y OpenAI Gym. En este entorno, se integra el robot ROSbot. Las simulaciones se realizan para evaluar el rendimiento de los algoritmos en tareas de navegación de robots, analizando diferentes métricas de desempeño.

Una vez hecho esto, se desarrollan documentos detallados sobre el proceso de instalación, configuración y ajuste de los entornos y algoritmos. Estos documentos sirven como una guía práctica para futuros trabajos en este campo. Finalmente, el análisis comparativo de los resultados permite identificar las fortalezas y debilidades de cada algoritmo, proponiendo posibles mejoras y futuras líneas de investigación.

Abstract

In the field of research and application of reinforcement learning techniques, the use of advanced algorithms for training mobile robots in simulated environments is common. Algorithms such as Q-Learning and Sarsa, are fundamental in reinforcement learning. The development of this Project aims to implement and compare these two algorithms to improve the performance of robots in specific tasks.

To achieve this, the first step is to configure the simulation environment using tools like Gazebo, ROS, and OpenAI Gym. In this environment, the ROSbot robot is integrated. Simulations are conducted to evaluate the performance of the algorithms in tasks of navigation, analyzing various performance metrics.

Subsequently, detailed documentation is developed on the installation, configuration, and adjustment processes of the environments and algorithms. These documents serve as a practical guide for future work in this field. Finally, the comparative analysis of the results identifies the strengths and weaknesses of each algorithm, proposing possible improvements and future lines of research.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

El campo de la robótica móvil ha experimentado un importante crecimiento en las últimas décadas, impulsado principalmente por los avances en la inteligencia artificial y la tecnología de sensores. Este crecimiento ha abierto nuevas posibilidades para la automatización y el desarrollo de sistemas autónomos, que pueden llevar a cabo tareas complejas en diversos entornos.

La robótica móvil es utilizada en una amplia gama de aplicaciones, tales como la exploración espacial, la agricultura, la logística o la manufactura. La capacidad de estos robots para navegar y realizar tareas de manera autónoma depende en gran medida de la tecnología de sensores y algoritmos de inteligencia artificial que permiten percibir, interpretar y reaccionar ante el entorno en tiempo real.

Un desafío interesante y complejo dentro de este campo es el control de múltiples robots móviles, el cual implica no solo la navegación autónoma de cada robot, sino también la colaboración entre ellos para lograr objetivos comunes. En este contexto, el control con obstáculos móviles debe tener en cuenta las lecturas del entorno y la posición de los otros robots para garantizar movimientos precisos y evitar colisiones.

1.1. Objetivos del trabajo

El objetivo principal de este trabajo es investigar y aplicar diversas técnicas de aprendizaje por refuerzo para entrenar agentes autónomos en entornos simulados, específicamente en el contexto de robots móviles. Este estudio se centrará en la implementación, comparación y análisis de dos algoritmos fundamentales de este tipo de aprendizaje: Q-Learning y Sarsa.

En primer lugar, se buscará comprender y explicar los fundamentos teóricos del aprendizaje por refuerzo, abarcando conceptos esenciales como el agente, entorno, estados, acciones, recompensas y el ciclo de interacción entre todos estos elementos.

En segundo lugar, se pretende implementar y ajustar los algoritmos Q-Learning y Sarsa en el entorno de simulación Gazebo, utilizando el middleware ROS (Robot Operating System) y la plataforma OpenAI Gym. Además, se evaluará el rendimiento de estos algoritmos en tareas específicas de navegación de robots mediante la simulación de uno y múltiples robots, analizando métricas de desempeño tales como la recompensa acumulada, la eficiencia de aprendizaje y la capacidad de adaptación.

Otro objetivo importante es comparar los resultados obtenidos de los diferentes algoritmos con el fin de identificar sus fortalezas y debilidades en contextos individuales y multi-robot, de cara a proponer posibles

mejoras.

Finalmente, se desarrollará una documentación detallada del proceso de instalación, configuración y ajuste de los entornos de simulación y los algoritmos de aprendizaje por refuerzo, con el objetivo de proporcionar una guía práctica para futuros trabajos en este campo.

1.2. Alcance del proyecto

El alcance de este proyecto abarca la implementación y evaluación de técnicas de aprendizaje por refuerzo en el contexto de robots móviles simulados en Gazebo.

En primer lugar, se aborda la configuración del entorno de simulación, lo que incluye la instalación y configuración de Gazebo, ROS y OpenAI Gym, así como la integración del robot ROSbot en el entorno de simulación.

En segundo lugar, se desarrollan e implementan los algoritmos de aprendizaje por refuerzo, que implica el análisis de los algoritmos Q-Learning y Sarsa y su adaptación para funcionar tanto en entornos de un solo robot como en múltiples robots. Posteriormente, se procede con el entrenamiento y evaluación de los robots, definiendo tareas específicas de navegación y analizando las métricas de desempeño obtenidas.

Finalmente, se documenta y analiza todo el proceso, lo que incluye la elaboración de informes detallados sobre la configuración, implementación y resultados obtenidos, así como la comparación de los algoritmos en términos de desempeño y eficiencia. Con base en estos hallazgos, se proponen mejoras y futuras líneas de investigación.

1.3. Estructura de la memoria

Este Trabajo de Fin de Grado está estructurado para ofrecer una visión completa y detallada del aprendizaje por refuerzo aplicado a la robótica móvil, abarcando desde los conceptos teóricos hasta la implementación práctica y el análisis de los resultados. A continuación, se describe cómo se organiza y qué contenido incluye cada capítulo:

En el capítulo 2 se profundiza en el aprendizaje por refuerzo, abordando su definición, principios teóricos, modelos, métodos de solución y el dilema de exploración-explotación, proporcionando una base sólida para su comprensión y aplicación.

El capítulo 3 está dedicado a los algoritmos Q-Learning y Sarsa, explicando sus fundamentos, funcionamiento, parámetros y aplicaciones, además de ofrecer una comparación para evaluar sus ventajas y desventajas.

El capítulo 4 se centra en las herramientas utilizadas en el proyecto, como Gazebo, ROS y OpenAI Gym, así como en su integración para crear el entorno de simulación necesario para el entrenamiento y evaluación de los robots.

El capítulo 5 detalla la configuración e integración de los robots en Gazebo, incluyendo la instalación, ajustes necesarios y la navegación de múltiples robots dentro del entorno simulado.

El capítulo 6 presenta los resultados obtenidos del entrenamiento de los robots utilizando Q-Learning y Sarsa, analizando su desempeño en términos de recompensa acumulada, eficiencia del aprendizaje y capacidad de adaptación en entornos individuales y con múltiples robots.

Finalmente, el capítulo 7 resume las conclusiones del análisis, comparando los resultados de los algoritmos y su rendimiento en distintos contextos, mientras que el capítulo 8 propone futuras líneas de investigación, como el desarrollo de Deep Q-Learning y la mejora de los algoritmos actuales.

2 APRENDIZAJE POR REFUERZO

El aprendizaje por refuerzo es un campo fundamental del aprendizaje automático, donde un agente aprende a tomar decisiones óptimas mediante la interacción continua con su entorno. A diferencia de otros paradigmas de aprendizaje como el supervisado y el no supervisado, el aprendizaje por refuerzo no depende de etiquetas explícitas en los datos. En su lugar, el agente recibe retroalimentación en forma de recompensas y penalizaciones, ajustando sus acciones para maximizar la recompensa acumulada a lo largo del tiempo, lo cual permite que el agente mejore su rendimiento basado en experiencias previas, promoviendo un aprendizaje adaptativo y continuo.

La versatilidad del aprendizaje por refuerzo se refleja en su amplia gama de aplicaciones prácticas. Desde superar el rendimiento humano en juegos complejos hasta optimizar la conducta de robots en entornos dinámicos, este tipo de aprendizaje demuestra su capacidad para desarrollar estrategias óptimas en diversas situaciones. Estas aplicaciones ponen en relevancia el potencial del aprendizaje por refuerzo para transformar diversas industrias mediante la implementación de agentes inteligentes capaces de aprender y adaptarse a través de la interacción continua con su entorno [1].

2.1. Introducción al aprendizaje por refuerzo

2.1.1 Definición y concepto básico

El aprendizaje por refuerzo es una rama del aprendizaje automático en la que un agente aprende a tomar decisiones mediante la interacción continua con su entorno. Este proceso se basa en un sistema de recompensas y penalizaciones que guían el aprendizaje del agente, donde el objetivo principal del agente es maximizar la recompensa acumulada a lo largo del tiempo, ajustando sus acciones basadas en las experiencias previas para mejorar su rendimiento futuro [3] [4].

2.1.2 Comparación con otros paradigmas de aprendizaje (supervisado y no supervisado)

El aprendizaje por refuerzo se distingue de otros paradigmas de aprendizaje automático, como el aprendizaje supervisado y no supervisado.

En el aprendizaje supervisado, un modelo se entrena utilizando un conjunto de datos etiquetados donde cada entrada está asociada a una salida deseada, mientras que el aprendizaje por refuerzo no necesita etiquetas

explícitas, ya que depende de la retroalimentación del entorno en forma de recompensas y penalizaciones.

Por otro lado, el aprendizaje no supervisado busca patrones o estructuras en datos no etiquetados, sin una guía específica sobre qué se considera correcto o incorrecto. En el aprendizaje por refuerzo, el foco está en la toma de decisiones óptimas a través de la interacción continua con el entorno y la maximización de la recompensa acumulada, lo cual implica un proceso de ensayo y error donde el agente mejora su política de acciones basada en las recompensas recibidas.

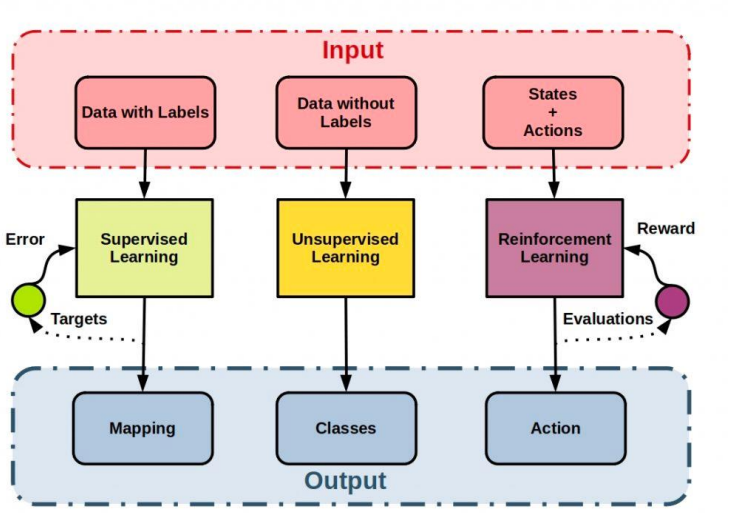


Figura 2-1. Comparación de paradigmas de aprendizaje automático[1]

2.1.3 Aplicaciones del aprendizaje por refuerzo

El aprendizaje por refuerzo tiene una amplia gama de aplicaciones prácticas. Una de las más conocidas es en el campo de los juegos, donde algoritmos de aprendizaje por refuerzo han alcanzado niveles de rendimiento superiores a los humanos en juegos complejos como el ajedrez. En el control de robots, se utiliza para desarrollar políticas de control óptimas que permiten a los robots realizar tareas complejas en entornos dinámicos [4].

En el sector financiero, se aplica en la optimización de carteras de inversión, donde el objetivo es maximizar las ganancias ajustadas por riesgo a lo largo del tiempo. También es comúnmente empleado en los sistemas de recomendación para mejorar la personalización de recomendaciones a usuarios en plataformas de contenido y comercio electrónico. Estas aplicaciones demuestran la versatilidad y potencia del aprendizaje por refuerzo para aprender estrategias óptimas a través de la interacción continua con su entorno.

2.2. Fundamentos del aprendizaje por refuerzo

2.2.1. Elementos clave: agente, entorno, acciones, estados y recompensas

En el aprendizaje por refuerzo, el agente interactúa con un entorno realizando acciones en diferentes estados y recibe recompensas como retroalimentación [1] [2].

Los elementos principales son:

- Agente: Es la entidad que toma decisiones y realiza acciones. Este agente debe tomar decisiones en tiempo real sobre qué hacer para optimizar su rendimiento en el entorno.
- Entorno: Es todo aquello con lo que el agente interactúa, proporcionando la base sobre la cual el agente observa y evalúa sus acciones y decisiones. Cada interacción del agente con el entorno proporciona nueva información que el agente puede usar para mejorar su estrategia.
- Acciones (A): Son los movimientos o decisiones que el agente puede realizar en el entorno. Estas

[1] https://pub.towardsai.net/introduction-to-reinforcement-learning-series-23492a319735

acciones determinan cómo el agente interactúa con el entorno y afectan directamente el resultado de cada episodio.

- Estados (S): Representan la situación actual del entorno en un momento dado. Los estados son muy importantes porque proporcionan el contexto necesario para que el agente tome decisiones informadas sobre qué acción realizar a continuación.
- Recompensas (R): Son la retroalimentación que el agente recibe después de realizar una acción, pudiendo ser positivas o negativas en función del resultado de la acción. Las recompensas son esenciales para guiar el proceso de aprendizaje del agente, ya que le indican qué acciones son beneficiosas y cuáles no lo son.

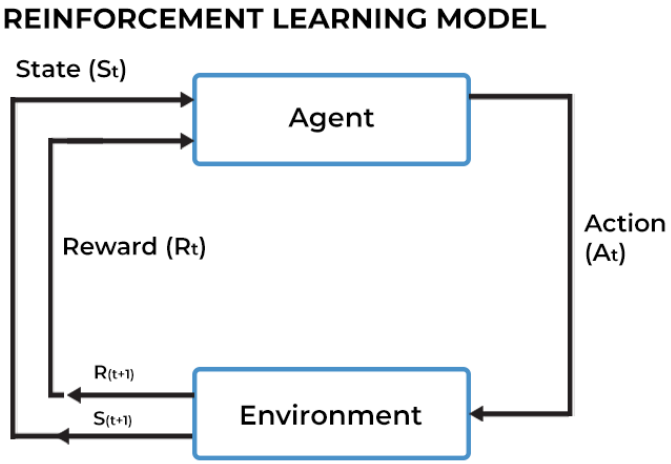


Figura 2-2. Ciclo de interacción del aprendizaje por refuerzo[2]

2.2.2. Ciclo de interacción agente-entorno

En el aprendizaje por refuerzo, el ciclo de interacción entre el agente y el entorno es fundamental para el proceso de aprendizaje. Este ciclo comienza con la observación del estado actual del entorno. En esta etapa, el agente recopila información sobre la situación presente del entorno, que puede incluir la posición de objetos, las condiciones ambientales o cualquier otra variable relevante que define el estado del entorno en ese momento específico. Esta observación es crítica en el proceso de aprendizaje, ya que proporciona la base sobre la cual el agente tomará decisiones posteriores [1] [2].

A continuación, el agente procede con la selección de una acción. Basado en una política previamente definida, que es una estrategia que el agente sigue para tomar decisiones, el agente determina cuál es la mejor acción para realizar en el estado observado. La política puede estar influenciada por la experiencia previa del agente, es decir, las acciones que han resultado en recompensas positivas en situaciones similares en el pasado. Este proceso de selección dirige el comportamiento del agente hacia la maximización de las recompensas.

Una vez que se ha seleccionado una acción, el siguiente paso es la ejecución de la acción por parte del agente, interfiriendo activamente en el entorno. La ejecución efectiva de la acción impacta directamente en el estado del entorno y, por lo tanto, en las recompensas posteriores.

Después de ejecutar la acción, el agente pasa a la fase de recepción de recompensa y nuevo estado. Como resultado de la acción realizada, el agente recibe una recompensa que puede ser positiva o negativa, dependiendo de si la acción acercó al agente a su objetivo. Simultáneamente, el entorno transita a un nuevo estado que refleja las consecuencias de la acción tomada. Esta etapa proporciona una retroalimentación inmediata al agente sobre la eficacia de su acción.

[2] https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-reinforcement-learning/

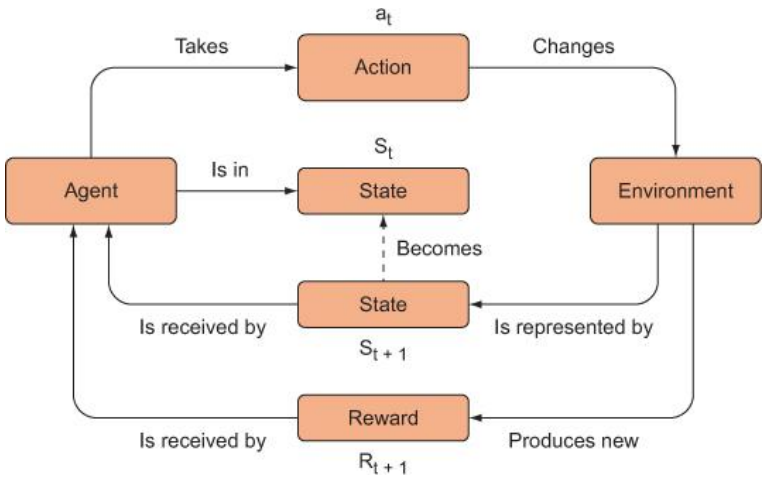


Figura 2-3. Ciclo de interacción en el aprendizaje por refuerzo[3]

Este ciclo se repite continuamente mientras el agente interactúa con el entorno. Con cada iteración, el agente utiliza la información recibida, es decir, el nuevo estado y la recompensa obtenida, para ajustar y mejorar su política. El objetivo es optimizar la toma de decisiones futuras para maximizar la recompensa acumulada a lo largo del tiempo. Este proceso de aprendizaje implica explorar diferentes acciones y estados para descubrir cuáles producen las mayores recompensas, y luego explotar este conocimiento para tomar decisiones que maximicen la recompensa acumulada. Esto permite que el agente aprenda de la experiencia, refinando su estrategia y mejorando su desempeño en tareas futuras.

En cada paso de este ciclo, la capacidad del agente para utilizar eficazmente la información recibida es de gran importancia para el éxito del aprendizaje por refuerzo. La mejora continua de la política del agente a través de la interacción con el entorno es lo que permite a los sistemas de aprendizaje por refuerzo desarrollar comportamientos complejos y efectivos en una variedad de aplicaciones.

2.3. Modelos de aprendizaje por refuerzo

En este apartado, se exploran los fundamentos matemáticos y conceptuales de los modelos de aprendizaje por refuerzo, centrándose en los procesos de decisión de Markov (MDP), que constituyen la base de muchos algoritmos de aprendizaje por refuerzo. A lo largo de este apartado, se detallan los componentes esenciales de un MDP, se explica la propiedad de Markov y se discute cómo estos conceptos facilitan la modelización y resolución de problemas de toma de decisiones secuenciales [5] [6].

2.3.1. Procesos de decisión de Markov

Los procesos de decisión de Markov son la base matemática del aprendizaje por refuerzo. Un MDP se define por un conjunto de estados S, un conjunto de acciones A, una función de transición de estados P y una función de recompensa R. Los MDPs asumen que las decisiones futuras dependen únicamente del estado actual y no del historial previo, lo que se conoce como la propiedad de Markov.

La propiedad de Markov se define formalmente como sigue:

Para una secuencia de estados S_t y una secuencia de acciones A_t, la probabilidad de transitar al siguiente estado S_{t+1} dado el estado actual S_t y la acción actual A_t es igual a la probabilidad de transitar al siguiente estado S_{t+1} dada la secuencia completa de estados y acciones anteriores. Matemáticamente, esto se expresa como

$$P(S_{t+1}|S_t, A_t) = P(S_{t+1}|S_0, S_1, \dots, S_t, A_0, A_1, \dots, A_t) \quad (2\text{-}1)$$

Donde:

- $P(S_{t+1}|S_t, A_t)$ es la probabilidad condicional de que el sistema transite al estado S_{t+1} en el tiempo $t +$

[3] https://livebook.manning.com/concept/deep-learning/objective

1 dado que el sistema está en el estado S_t en el tiempo t y se toma la acción A_t en el tiempo t.

- P representa la probabilidad de un evento.
- S_{t+1} es el estado del sistema en el tiempo $t+1$.
- S_t es el estado del sistema en el tiempo t.
- A_t es la acción tomada en el tiempo t.

- $P(S_{t+1}|S_0, S_1, \dots, S_t, A_0, A_1, \dots, A_t)$ es la probabilidad condicional de que el sistema transite al estado S_{t+1} en el tiempo $t+1$ dada la secuencia completa de estados $S_0, S_1, \dots, S_t$ y la secuencia completa de acciones $A_0, A_1, \dots, A_t$ hasta el tiempo t.
  - $S_0, S_1, \dots, S_t$ representan la secuencia de estados desde el tiempo 0 hasta el tiempo t.
  - $A_0, A_1, \dots, A_t$ representan la secuencia de estados desde el tiempo 0 hasta el tiempo t.

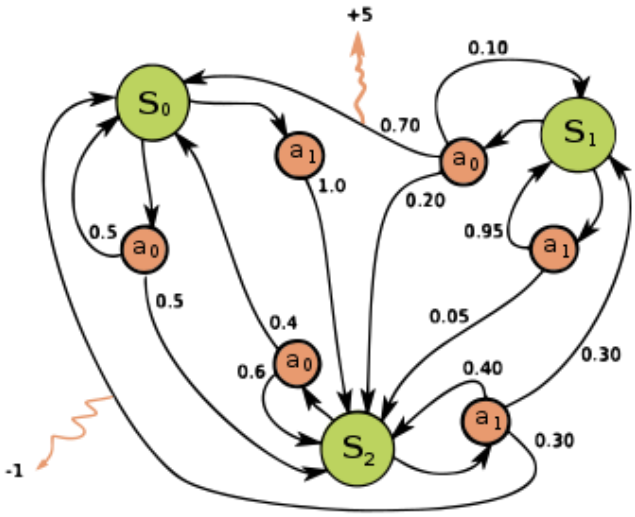


Figura 2-4. Diagrama de un proceso de decisión de Markov.[4]

La figura 2-4 representa un proceso de decisión de Markov que modela un sistema donde un agente toma decisiones secuenciales. Los estados del sistema están representados por los círculos verdes etiquetados como S_0, S_1 y S_2. Estos estados son situaciones específicas en las que puede encontrarse el sistema.

El agente puede tomar diversas acciones, representadas por los círculos naranjas etiquetados como a_0 y a_1. Cada acción influye en el sistema y provoca una transición de un estado a otro. Las flechas indican estas transiciones; por ejemplo, al tomar la acción a_0 desde el estado S_0, el sistema puede moverse a los estados S_0 o S_2 con una probabilidad del 50%.

Las líneas onduladas en la figura simbolizan las recompensas asociadas con las transiciones entre estados. Estas recompensas son los valores que el agente recibe al ejecutar ciertas acciones y cambiar de estado, y lo guían en la toma de decisiones.

Gracias a la propiedad de Markov, se puede simplificar enormemente el modelado de sistemas dinámicos, ya que no es necesario mantener un historial completo de todos los estados y acciones anteriores; solo se necesita conocer el estado y la acción actuales. Esto permite definir la matriz de transición de estado utilizando únicamente el estado y la acción actuales, lo que hace que los cálculos de los algoritmos de aprendizaje sean más eficientes y manejables.

Los MDPs son una representación matemática potente para modelar problemas donde la toma de decisiones es secuencial y las acciones influyen en el entorno de manera probabilística.

[4] https://es.wikipedia.org/wiki/Proceso_de_decisi%C3%B3n_de_M%C3%A1rkov

2.3.2. Políticas y funciones de valor

En el contexto de los procesos de decisión de Markov, una política es una estrategia que guía al agente en la toma de decisiones sobre qué acciones realizar en los diferentes estados del entorno. Matemáticamente, una política, denotada como π, es una función que asigna una acción A_t a cada estado S_t. Esto significa que, dado un estado S_t, la política π determina la acción A_t que el agente debe tomar [6].

$$\pi(S_t) = A_t \tag{2-2}$$

Existen políticas deterministas y estocásticas, que son dos estrategias fundamentales para la toma de decisiones del agente.

- Políticas deterministas: Una política determinista selecciona siempre la misma acción para un estado dado. Formalmente, se denota como $\pi(S_t)$, donde π es una función que asigna a cada estado S_t una acción específica A_t. Este enfoque es sencillo y eficiente en entornos donde la incertidumbre es mínima y la variabilidad en las decisiones no es crítica.
- Políticas estocásticas: A diferencia de las deterministas, las políticas estocásticas seleccionan acciones basadas en una distribución de probabilidad. Se denotan como $\pi(A_t|S_t)$ donde π es una función que asigna una probabilidad a cada acción A_t en un estado S_t. Este método es más robusto en entornos donde la incertidumbre es significativa y es beneficioso explorar múltiples acciones para un mismo estado.

En el contexto del aprendizaje por refuerzo, es esencial evaluar la calidad de las decisiones que toma un agente. Para ello, se utilizan las funciones de valor, que permiten estimar las recompensas futuras esperadas. Estas funciones no solo ayudan a entender cuánto beneficio puede esperar un agente al encontrarse en un determinado estado, sino también al tomar acciones específicas dentro de ese estado. Se distinguen dos tipos principales: funciones de valor de estado y funciones de valor de acción.

2.3.2.1. Función de valor de estado

La función de valor de estado, denotada como $V(S_t)$, representa el valor esperado de las recompensas futuras que pueden obtenerse desde un estado S_t al seguir una política π. Es una medida de cuán bueno es un estado dado si el agente sigue la política π [1].

Se define matemáticamente como:

$$V^{\pi}(S_t) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \,|S_0 = S_t\right] \tag{2-3}$$

En esta fórmula:

- $\mathbb{E}_{\pi}$ indica el valor esperado bajo la política π.
- $\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t)$ es la suma de las recompensas futuras descontadas.
- γ es el factor de descuento, un valor entre 0 y 1 que reduce el valor de las recompensas futuras para reflejar su importancia decreciente a medida que se alejan en el tiempo.
- $S_0 = S_t$ indica que el cálculo del valor esperado comienza desde el estado S_t.

2.3.2.2. Función de valor de acción

La función de valor de acción, denotada como $Q(S_t, A_t)$ representa el valor esperado de las recompensas futuras que pueden obtenerse desde un estado S_t al tomar una acción A_t específica y luego seguir la política π [1].

Se define matemáticamente como:

$$Q^{\pi}(S_t, A_t) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \,|S_0 = S_t, A_0 = A_t\right] \tag{2-4}$$

En esta fórmula:

- $\mathbb{E}_\pi$ indica el valor esperado bajo la política π.
- $\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t)$ es la suma de las recompensas futuras descontadas.
- γ es el factor de descuento.
- $S_0 = S_t$ indica que el cálculo del valor esperado comienza desde el estado S_t.
- $A_0 = A_t$ indica que la primera acción tomada es A_t.

2.3.3. Ecuación de Bellman

La ecuación de Bellman es una fórmula matemática fundamental en el aprendizaje por refuerzo. Su propósito principal es proporcionar una relación recursiva para las funciones de valor, lo que ayuda a encontrar la política óptima que maximiza la recompensa acumulada a lo largo del tiempo [1][2].

Esta ecuación descompone el valor de un estado en términos de la recompensa inmediata y el valor descontado del próximo estado:

- Recompensa inmediata: La recompensa que el agente recibe inmediatamente al estar en un estado específico y tomar una acción específica.
- Valor descontado del próximo estado: El valor esperado del próximo estado ajustado por un factor de descuento γ, que mide la importancia de las recompensas futuras en comparación con las inmediatas.

2.3.3.1. Aplicación a la función de valor de estado

La ecuación de Bellman se aplica a la función de valor de estado óptima $V^*(S_t)$, que considera el valor de un estado S_t maximizando el valor futuro esperado mediante la mejor política posible. Matemáticamente se expresa como:

$$V^*(S_t) = \max_{A_t} \sum_{S_{t+1}} P(S_{t+1}|S_t, A_t)[R(S_t, A_t, S_{t+1}) + \gamma V^*(S_{t+1})] \quad (2\text{-}5)$$

Donde:

- $V^*(S_t)$ es el valor óptimo del estado S_t.
- $\max_{A_t}$ indica que se toma la acción A_t que maximiza el valor.
- $\sum_{S_{t+1}}$ es la suma sobre todos los posibles estados S_{t+1} a los que se puede transitar.
- $P(S_{t+1}|S_t, A_t)$ es la probabilidad de transitar al estado S_{t+1} desde el estado S_t al tomar la acción A_t.
- $R(S_t, A_t, S_{t+1})$ es la recompensa recibida al transitar del estado S_t al estado S_{t+1} mediante la acción A_t.
- γ es el factor de descuento que valora las recompensas futuras.

Esta fórmula describe cómo evaluar el valor de un estado en función de las recompensas inmediatas y las recompensas futuras esperadas, optimizando el proceso de decisión. Al maximizar el valor esperado mediante la mejor política posible, la ecuación de Bellman para la función de valor de estado óptima permite encontrar la estrategia óptima para un agente, lo que es esencial para resolver problemas de toma de decisiones secuenciales.

2.3.3.2. Aplicación a la función de valor de acción

La ecuación de Bellman también se aplica a la función de valor de acción óptima $Q^*(S_t, A_t)$, que considera el valor de tomar una acción específica A_t en un estado S_t, maximizando el valor esperado de las futuras recompensas descontadas siguiendo la mejor política posible:

$$Q^*(S_t, A_t) = \sum_{S_{t+1}} P(S_{t+1}|S_t, A_t)\left[R(S_t, A_t, S_{t+1}) + \gamma \max_{A_{t+1}} Q^*(S_{t+1}, A_{t+1}))\right] \quad (2\text{-}6)$$

Donde:

- $Q^*(S_t, A_t)$ es el valor óptimo de tomar la acción A_t en el estado S_t.
- $\max_{A_{t+1}}$ indica que se toma la mejor acción A_{t+1} en el próximo estado S_{t+1}.

La solución de estas ecuaciones permite encontrar las políticas óptimas y las funciones de valor. Estas ecuaciones forman la base para diseñar algoritmos de aprendizaje por refuerzo, permitiendo a los agentes aprender y tomar decisiones óptimas en entornos complejos.

2.4. Métodos de solución en aprendizaje por refuerzo

En el campo del aprendizaje por refuerzo, los métodos de solución se dividen principalmente en dos categorías: basados en modelos y sin modelos [7] [8] [9].

Los métodos basados en modelos utilizan un modelo del entorno para predecir los resultados de las acciones antes de ejecutarlas, lo que puede reducir significativamente el número de interacciones necesarias entre el agente y el entorno. Dentro de esta categoría, se encuentran estrategias como la búsqueda de árboles, la iteración de políticas y la iteración de valor, cada una con sus propias características y aplicaciones.

Por otro lado, los métodos sin modelos no requieren un conocimiento previo de las transiciones del entorno, lo que los hace más flexibles y fáciles de implementar en situaciones dinámicas o desconocidas. Ejemplos destacados de esta categoría son las diferencias temporales, que ajustan los valores de los estados basándose en la diferencia entre los valores consecutivos, y algoritmos específicos como Q-Learning y Sarsa.

2.4.1. Métodos basados en el modelo

El aprendizaje por refuerzo basado en modelos se caracteriza por la utilización de un modelo del entorno que permite prever el resultado de las acciones del robot antes de ejecutarlas [10].

Una de las principales ventajas de este método es que requiere un menor número de interacciones entre el robot y el entorno. Al tener la capacidad de prever las consecuencias de sus acciones, el robot puede aprender y planificar de manera más eficiente, lo que se traduce en una convergencia más rápida hacia una solución óptima [14].

Sin embargo, esta metodología también presenta desventajas significativas. La dependencia de los modelos de transición significa que la precisión del aprendizaje está estrechamente ligada a la exactitud del modelo del entorno. Si el modelo es impreciso o está mal definido, el rendimiento del algoritmo puede verse gravemente afectado, conduciendo a decisiones subóptimas y, en última instancia, a un aprendizaje ineficaz.

Estos métodos se dividen en tres estrategias principales:

2.4.1.1. Búsqueda de árboles

La búsqueda de árboles es una técnica que explora las posibles acciones y sus resultados en forma de árbol. Este método se basa en construir un árbol de decisiones donde cada nodo representa un estado y cada rama una acción. Al evaluar las posibles secuencias de acciones y sus consecuencias, el agente puede tomar decisiones informadas sobre cuál es la mejor acción para realizar en cada estado. Esta técnica es especialmente útil en entornos donde se puede prever una cantidad limitada de pasos hacia el futuro, permitiendo al agente planificar sus movimientos de manera eficiente [7].

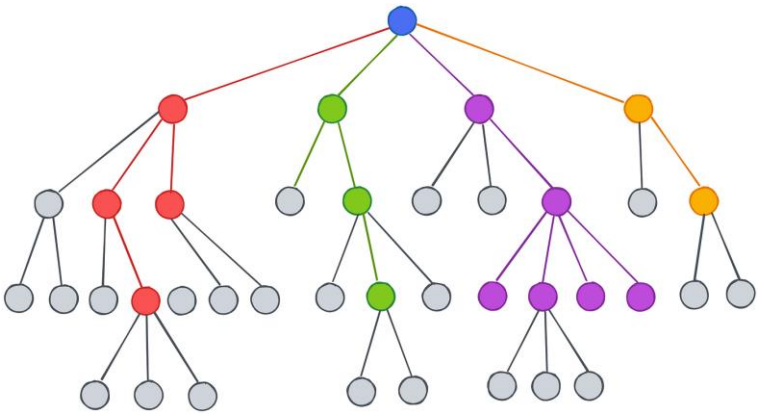


Figura 2-5. Estructura de un árbol de decisión en aprendizaje automático[5]

La figura 2-5 ilustra la estructura de un árbol de decisión en el contexto de la búsqueda de árboles en aprendizaje automático. En este árbol, cada nodo representa un estado del entorno y cada rama representa una acción posible que el agente puede tomar. El árbol se expande desde un nodo raíz en la parte superior, bifurcándose en varios nodos hijos que representan las diferentes decisiones posibles y sus subsecuentes estados.

A medida que se desciende por el árbol, las decisiones se vuelven más específicas, llevando a estados finales o nodos hoja, que se encuentran en la parte inferior del árbol. Los nodos están coloreados de manera diferente, lo que indica la clasificación o la decisión resultante en un problema específico. Los nodos grises en los extremos del árbol representan los resultados finales de las secuencias de decisiones.

2.4.1.2. Iteración de políticas

La iteración de políticas es un proceso iterativo que implica dos etapas: evaluación de la política y mejora de la política. En la fase de evaluación, se calcula el valor esperado de cada estado bajo la política actual, utilizando la ecuación de Bellman para estimar los valores. En la fase de mejora, se ajusta la política para maximizar el valor esperado, eligiendo la mejor acción en cada estado basado en los valores calculados. Este proceso se repite hasta que la política converge a una política óptima [1].

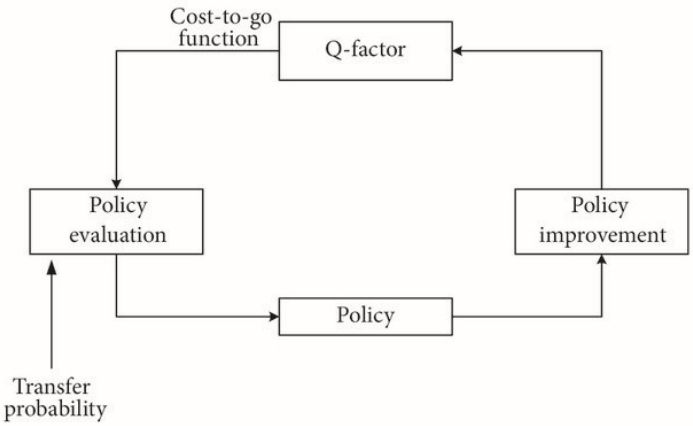


Figura 2-6. Iteración de política en el aprendizaje por refuerzo[6]

La figura 2-6 representa el ciclo de iteración de política en el aprendizaje por refuerzo. Este ciclo se compone de varios pasos interrelacionados que optimizan la política del agente.

Se comienza con una política inicial, que puede ser aleatoria o basada en algún conocimiento previo, definiendo las acciones que el agente debe tomar en cada estado del entorno. En la etapa de evaluación de política, se calcula el valor esperado de cada estado representando la recompensa esperada siguiendo la política actual.

[5] https://medium.com/intro-to-artificial-intelligence/decision-tree-learning-e153b5b4ecdf
[6] https://www.researchgate.net/figure/Illustration-of-policy-iteration-in-reinforcement-learning_fig1_324725969

Luego, en la mejora de política, se actualiza la política para maximizar la recompensa esperada, seleccionando acciones que optimicen el valor esperado en cada estado. El Q-factor estima el valor de tomar una acción específica en un estado determinado, ayudando a evaluar y mejorar la política. La probabilidad de transferencia indica la transición entre estados en función de la acción tomada, utilizada para calcular la recompensa esperada y ajustar la política.

Este ciclo de iteración continúa hasta que la política converge a una solución óptima, maximizando la recompensa esperada en cada estado del entorno.

2.4.1.3. Iteración de valor

La iteración de valor es similar a la iteración de políticas, pero se enfoca directamente en la actualización de los valores de los estados. En lugar de evaluar una política fija, este método actualiza los valores de los estados iterativamente usando la ecuación de Bellman, sin una política explícita. Una vez que los valores convergen, se puede derivar una política óptima eligiendo las acciones que maximicen el valor en cada estado. Este método es eficiente y garantiza la convergencia a los valores óptimos [1].

2.4.2. Métodos sin modelo

El aprendizaje por refuerzo sin modelos no requiere un conocimiento previo de las transiciones del entorno. Esto hace que estos algoritmos sean más flexibles y fáciles de implementar, ya que pueden ser aplicados directamente en entornos desconocidos sin necesidad de una fase de modelado previa. Esta simplicidad y adaptabilidad son ventajas importantes, especialmente en aplicaciones donde el entorno es dinámico o difícil de modelar con precisión [10].

No obstante, los métodos sin modelos también tienen sus inconvenientes. La convergencia del aprendizaje suele ser más lenta debido a que dependen de la prueba y el error para descubrir la mejor política de acción, lo cual también puede ser costoso en términos de tiempo y recursos. Además, el alto número de interacciones necesarias puede llevar a un desgaste del robot, aumentando los costes de mantenimiento y el riesgo de daños físicos, especialmente en entornos hostiles o impredecibles.

Estos métodos incluyen:

2.4.2.1. Diferencias temporales

El aprendizaje por diferencias temporales es una técnica que actualiza los valores de los estados basándose en la diferencia entre los valores consecutivos. Este método combina las ideas del aprendizaje supervisado y el aprendizaje por refuerzo, ajustando los valores de los estados usando la recompensa inmediata y el valor descontado del siguiente estado [10].

Una fórmula comúnmente utilizada en este método es:

$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_t + \gamma V(S_{t+1}) - V(S_t)\big) \tag{2-7}$$

Donde:

- $V(S_t)$ es el valor del estado actual S_t. Representa la estimación de la recompensa total esperada que se puede obtener desde el estado S_t siguiendo una política determinada.
- α es la tasa de aprendizaje. Es un parámetro entre 0 y 1 que determina cuánto de la nueva información (la diferencia entre la recompensa esperada y la actual) se incorpora a la estimación del valor del estado. Un α más alto significa que se da más peso a la nueva información.
- R_t es la recompensa inmediata recibida al tomar una acción en el estado S_t y pasar al estado siguiente S_{t+1}. Es la recompensa directa obtenida de la acción.
- γ es el factor de descuento. Es un parámetro entre 0 y 1 que determina la importancia de las recompensas futuras. Un γ más cercano a 1 significa que las recompensas futuras son casi tan importantes como las recompensas inmediatas, mientras que un γ más cercano a 0 hace que las recompensas futuras sean

menos importantes.

- $V(S_{t+1})$ es el valor del estado siguiente S_{t+1}. Es la estimación de la recompensa total esperada desde el nuevo estado S_{t+1} después de tomar una acción desde el estado S_t.

Esta fórmula representa matemáticamente la regla de actualización del valor del estado S_t. Se calcula la diferencia entre la recompensa obtenida más la recompensa esperada del nuevo estado ($R_t + \gamma V(S_{t+1})$) y la recompensa esperada del estado actual $V(S_t)$. Esta diferencia se multiplica por la tasa de aprendizaje α y se suma al valor actual $V(S_t)$ para obtener el nuevo valor actualizado del estado S_t.

Q-Learning y Sarsa están directamente relacionados con el concepto de diferencias temporales, ya que ambos algoritmos utilizan este enfoque para actualizar sus estimaciones de valor. Las diferencias temporales permiten a estos algoritmos ajustar continuamente las estimaciones de valor de las acciones en función de la diferencia entre las recompensas inmediatas y las estimaciones de valor futuro. Este método proporciona una forma eficiente y efectiva de aprendizaje, permitiendo a los agentes mejorar sus políticas con cada interacción en el entorno.

En el contexto de Q-Learning, el algoritmo actualiza sus valores Q utilizando la máxima recompensa esperada del siguiente estado, lo que lo clasifica como un algoritmo off-policy. Esto significa que puede aprender la política óptima independientemente de las acciones actuales del agente, enfocándose siempre en las mejores recompensas posibles a futuro. Por otro lado, Sarsa es un algoritmo on-policy que actualiza sus valores Q basándose en la acción real seleccionada por la política en uso, lo que permite al agente aprender una política coherente con su comportamiento actual en tiempo real.

En el siguiente capítulo se profundizará en estos dos algoritmos. Se explicarán en detalle sus mecanismos de actualización, cómo implementan la técnica de diferencias temporales y se discutirán ejemplos prácticos de su aplicación.

2.5. Exploración y explotación

En el aprendizaje por refuerzo, uno de los principales desafíos es equilibrar la exploración y la explotación del entorno. La exploración permite a los agentes probar nuevas acciones y descubrir estrategias potencialmente mejores, mientras que la explotación se centra en utilizar el conocimiento actual para maximizar las recompensas inmediatas [1] [5].

A lo largo del tiempo, se han desarrollado varias estrategias para gestionar este dilema, optimizando el rendimiento de los agentes al combinar de manera efectiva la exploración y la explotación.

2.5.1. Dilema exploración-explotación

El dilema exploración-explotación se refiere a la decisión que debe tomar un agente entre dos opciones:

- Explotación: Utilizar el conocimiento actual para maximizar la recompensa inmediata. Esto implica que el agente selecciona las acciones que ya sabe que proporcionan buenas recompensas basándose en experiencias pasadas.
- Exploración: Probar nuevas acciones que podrían no ser óptimas en el corto plazo pero que podrían proporcionar una mayor recompensa a largo plazo si se descubren mejores estrategias.

Este dilema es crítico, ya que un agente que solo explota su conocimiento actual puede quedar atrapado en una estrategia subóptima, mientras que un agente que solo explora nunca aprovechará completamente lo que ha aprendido. El equilibrio adecuado entre exploración y explotación permite al agente mejorar continuamente su política y maximizar su recompensa acumulada a lo largo del tiempo.

2.5.2. Estrategias de exploración

Para abordar el dilema exploración-explotación, se han desarrollado varias estrategias que permiten a los agentes explorar y explotar de manera equilibrada. Aquí se describen tres estrategias comunes:

- Epsilon-greedy: Esta estrategia implica que, con una alta probabilidad (1 - ε), el agente selecciona la

mejor acción conocida (explotación), pero con una pequeña probabilidad ε, el agente elige una acción al azar (exploración). Esta técnica asegura que el agente no se quede atrapado en una estrategia subóptima y tenga la oportunidad de descubrir nuevas y potencialmente mejores acciones. La probabilidad ε suele disminuir con el tiempo a medida que el agente adquiere más conocimiento sobre el entorno [10].

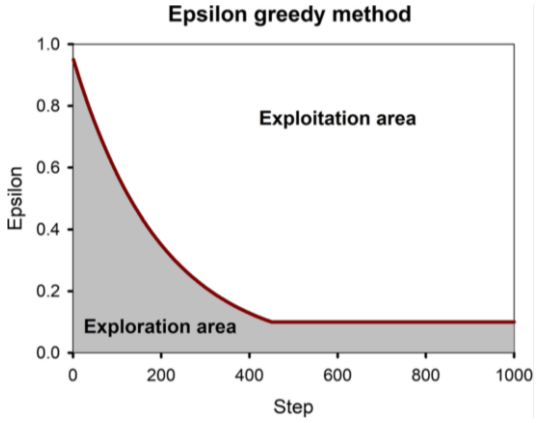


Figura 2-7. Reducción del valor de épsilon en el método ε-Greedy[7]

- Softmax: En la estrategia softmax, las acciones se seleccionan de acuerdo con una distribución de probabilidad que favorece las acciones con valores más altos. En lugar de seleccionar la acción con el valor más alto de forma determinista, se asigna una probabilidad a cada acción en función de su valor estimado, y las acciones se seleccionan de acuerdo con estas probabilidades. Esto permite una exploración más sutil y continua, ya que todas las acciones tienen una probabilidad positiva de ser seleccionadas, pero las mejores acciones son seleccionadas con mayor frecuencia [2].
- Upper Confidence Bound (UCB): La estrategia UCB selecciona acciones basándose en una combinación del valor estimado de la acción y la incertidumbre asociada a esa estimación. La idea es elegir la acción que tiene el mayor valor potencial según la información disponible y la cantidad de veces que se ha seleccionado esa acción anteriormente. Esta estrategia equilibra automáticamente la exploración y la explotación al priorizar las acciones menos exploradas, pero potencialmente valiosas, proporcionando una estrategia teóricamente sólida para maximizar la recompensa a largo plazo [1].

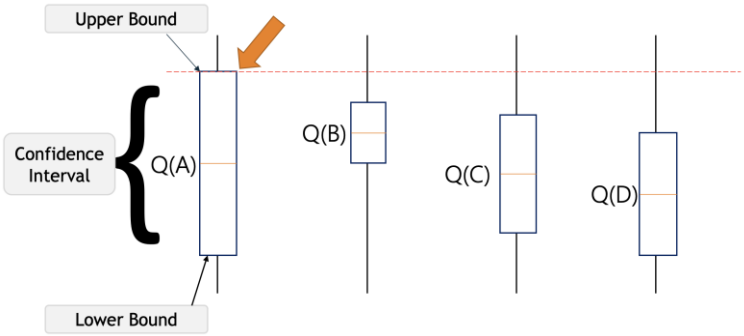


Figura 2-8. Selección de acción basada en el Upper Confidence Bound en valores Q[8]

La figura 2-8 muestra cómo se aplican los intervalos de confianza en los valores Q de diferentes acciones (A, B, C, D). Cada caja representa el intervalo de confianza de una acción específica, indicando los límites superior e inferior de las estimaciones. La estrategia UCB selecciona la acción con el límite superior más alto, resaltado en la figura con una flecha y el término "Upper Bound". Este método asegura que las acciones con alta incertidumbre y potencialmente alto valor sean seleccionadas para su exploración, equilibrando así la exploración y la explotación de manera eficiente.

[7] https://www.researchgate.net/figure/Epsilon-greedy-method-At-each-step-a-random-number-is-generated-by-the-model-If-the_fig2_334741451
[8] https://www.geeksforgeeks.org/upper-confidence-bound-algorithm-in-reinforcement-learning/

3 ALGORITMOS DE APRENDIZAJE POR REFUERZO

Dentro del campo del aprendizaje por refuerzo, se analizan tres algoritmos fundamentales del aprendizaje por refuerzo: Q-Learning y Sarsa. Estos algoritmos son esenciales para la implementación de agentes que aprendan a tomar decisiones óptimas a través de la interacción con su entorno.

Q-Learning es un algoritmo que permite a los agentes aprender políticas óptimas actualizando iterativamente los valores de acción basados en la ecuación de Bellman. Este método es efectivo en una variedad de aplicaciones, incluyendo la navegación de robots y la optimización de procesos industriales.

Sarsa se distingue de Q-Learning por ser un método on-policy, donde las actualizaciones de las políticas se basan en las acciones realmente tomadas por el agente, lo que puede llevar a una mayor estabilidad en ciertos entornos. Es útil en sistemas donde las decisiones seguras son cruciales, como en robótica.

En conjunto, estos algoritmos representan herramientas poderosas para la toma de decisiones óptimas en sistemas autónomos, cada uno adecuado para diferentes niveles de complejidad y tipos de entornos. A pesar de los desafíos comunes, como el equilibrio entre exploración y explotación y la escalabilidad, siguen siendo fundamentales en el avance del aprendizaje por refuerzo.

3.1. Q-Learning

El Q-Learning es un algoritmo de aprendizaje por refuerzo que permite a los agentes aprender políticas óptimas mediante la actualización iterativa de valores de acción. Este método ha demostrado ser eficaz en una variedad de aplicaciones, desde la navegación de robots hasta la optimización de procesos industriales y estrategias en videojuegos [10] [11].

3.1.1 Fundamentos del Q-Learning

Q-Learning es un algoritmo off-policy que busca aprender la función de valor óptima, $Q(S_t, A_t)$, sin seguir una política específica. Esto significa que el agente puede aprender de transiciones observadas, independientemente de la política utilizada para generar esas transiciones [8].

Este método actualiza los valores Q usando la siguiente regla:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma \max_{A_{t+1}} Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right) \quad (3\text{-}1)$$

Donde:

- $Q(S_t, A_t)$ es el valor actual de la función Q para el estado S_t y la acción A_t. Representa la estimación de la recompensa acumulada que el agente espera obtener a partir de realizar la acción A_t en el estado S_t.
- α es la tasa de aprendizaje.
- R_{t+1} es la recompensa recibida tras realizar la acción A_t en el estado S_t.
- γ es el factor de descuento que pondera las recompensas futuras.
- $\max_{A_{t+1}} Q(S_{t+1}, A_{t+1})$ es el valor máximo estimado de la función Q para el siguiente estado S_{t+1} y todas las posibles acciones A_{t+1}.

El objetivo del Q-Learning es iterar sobre esta ecuación para ajustar los valores Q utilizando las recompensas inmediatas y las estimaciones de los valores futuros, de manera que el agente pueda aprender una política óptima que maximice las recompensas acumuladas en el tiempo.

Una característica importante del Q-Learning es su capacidad de garantizar la convergencia a la función de valor óptima Q^*, incluso cuando el agente explora el entorno de manera no óptima. Esto se debe a su naturaleza off-policy, que le permite aprender de las experiencias generadas por diferentes políticas, incluidas aquellas que no son óptimas.

El proceso de convergencia de Q-Learning está respaldado por ciertas condiciones matemáticas. Los valores Q se almacenan en una tabla de consulta que cubre todas las combinaciones de estados y acciones posibles, asegurando una referencia completa del espacio de estado-acción. Además, la tasa de aprendizaje decreciente es fundamental, satisfaciendo las siguientes condiciones:

$$0 \leq \alpha_t(S_t, A_t) \leq 1 \quad (3\text{-}3)$$

$$\sum_t \alpha_t(S_t, A_t) = \infty \quad (3\text{-}4)$$

$$\sum_t {\alpha_t}^2(S_t, A_t) < \infty \quad (3\text{-}5)$$

Esto permite que el impacto de nuevas experiencias disminuya con el tiempo, estabilizando los valores Q. Asimismo, es necesario que las recompensas recibidas sean finitas para evitar que las actualizaciones de Q diverjan, manteniendo el proceso de aprendizaje controlado y dirigido hacia la convergencia.

3.1.2 Algoritmo de Q-Learning

El algoritmo de Q-Learning se implementa mediante un proceso iterativo que sigue los siguientes pasos:

1. Inicializar la función de valor Q de manera arbitraria.
2. Para cada episodio, repetir hasta alcanzar un estado terminal:
    a. Seleccionar una acción A_t en el estado S_t utilizando una política basada en Q (por ejemplo, ε-Greedy).
    b. Ejecutar la acción A_t y observar la recompensa R_{t+1} y el nuevo estado S_{t+1}.
    c. Actualizar $Q(S_t, A_t)$ utilizando la ecuación de Bellman.
    d. Establecer el nuevo estado S_{t+1} como el estado actual S_t.
3. Repetir los pasos anteriores hasta que los valores Q converjan a valores óptimos.

La tabla de valores Q es una tabla que el algoritmo de Q-Learning utiliza para almacenar los valores Q de cada par estado-acción. Cada fila de la tabla representa un estado del entorno y cada columna representa una posible acción. El valor en la intersección de una fila y una columna $Q(S_t, A_t)$ indica el valor estimado de tomar la acción A_t en el estado S_t. La tabla de valores Q se actualiza iterativamente a medida que el agente aprende de sus interacciones con el entorno, ajustando los valores Q para reflejar mejor la recompensa esperada de cada acción en cada estado [12].

Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	327	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	0	0	0	0	0	0

Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

Figura 3-1. Evolución de la tabla de valores Q en el algoritmo Q-Learning: inicialización y tras entrenamiento[9]

La figura 3-1 muestra la evolución de la tabla de valores Q en el algoritmo Q-Learning, comparando su estado inicial y su estado tras el entrenamiento. En la tabla inicializada, todos los valores de las acciones para cada estado se establecen en cero, indicando que no se ha adquirido ninguna información sobre las recompensas asociadas a las acciones en esos estados.

Después del proceso de entrenamiento, la tabla muestra valores actualizados para las acciones en diversos estados, reflejando el conocimiento adquirido por el agente sobre la mejor acción a tomar en cada estado para maximizar la recompensa acumulada a largo plazo. Estos valores indican la calidad esperada de las acciones, permitiendo al agente tomar decisiones informadas basadas en su experiencia.

3.1.3 Parámetros involucrados

En el algoritmo de Q-Learning intervienen tres parámetros que pueden ser modificados en función del modo de aprendizaje que se quiera obtener por parte del agente. Estos parámetros son alfa (α), gamma (γ) y épsilon (ε) [12].

Alfa (α): El parámetro alfa, también conocido como la tasa de aprendizaje, determina cuánto de la nueva información sobrescribe la antigua. Un valor de α de 0 significa que el agente no aprende nada nuevo y solo utiliza el conocimiento previo. Por otro lado, un α de 1 implica que el agente solo considera la información más reciente, ignorando todo conocimiento anterior.

Gamma (γ): El parámetro gamma, conocido como el factor de descuento, determina la importancia de las recompensas futuras. Un γ de 0 hace que el agente sea miope, solo considerando recompensas inmediatas. Un γ cercano a 1 hace que el agente valore las recompensas a largo plazo, mientras que un γ igual o mayor a 1 puede

[9] https://en.wikipedia.org/wiki/Q-learning#

llevar a valores de acción divergentes, por lo que generalmente se usa un valor ligeramente menor que 1.

Épsilon (ε): El parámetro épsilon se utiliza en estrategias de exploración-explotación, específicamente en la política ε-Greedy. Determina la probabilidad de que el agente explore nuevas acciones en lugar de explotar el conocimiento actual. Un ε alto fomenta la exploración, mientras que un ε bajo favorece la explotación de las acciones conocidas que proporcionan altas recompensas.

3.1.4 Aplicaciones de Q-Learning

El Q-Learning ha sido aplicado exitosamente en una variedad de campos, demostrando su flexibilidad y efectividad para resolver problemas complejos de toma de decisiones. A continuación, se describen algunas de las aplicaciones prácticas más destacadas [11].

En el campo de la robótica, el Q-Learning se ha utilizado extensivamente para enseñar a los robots a navegar en entornos desconocidos y dinámicos. Mediante este algoritmo, los robots pueden aprender a evitar obstáculos y a encontrar rutas óptimas hacia sus objetivos. Por ejemplo, en un entorno de fábrica, un robot equipado con Q-Learning puede identificar la mejor manera de moverse entre diferentes estaciones de trabajo mientras evita colisiones con otros robots y obstáculos. Este aprendizaje permite a los robots adaptarse a cambios en el entorno en tiempo real, mejorando así su eficiencia y autonomía.

En la industria manufacturera y de producción, el Q-Learning se aplica para optimizar procesos y mejorar la eficiencia operativa. Por ejemplo, en una línea de ensamblaje, el algoritmo puede ser utilizado para ajustar parámetros de las máquinas en tiempo real con el objetivo de maximizar la producción y minimizar el desperdicio. Además, puede ayudar en la gestión de energía, ajustando el uso de recursos energéticos para reducir costes y mejorar la sostenibilidad. La capacidad del Q-Learning para adaptarse y optimizar continuamente en función de las recompensas recibidas lo convierte en una herramienta valiosa para la automatización y el control de calidad en entornos industriales.

3.2. Sarsa

Sarsa (State-Action-Reward-State-Action) es un algoritmo de aprendizaje por refuerzo on-policy que actualiza las estimaciones de valor basándose en la política actual del agente [10] [11].

3.2.1. Definición y diferencias con Q-Learning

Sarsa es un algoritmo de aprendizaje por refuerzo similar a Q-Learning, pero con una diferencia clave en la forma en que actualiza las políticas y selecciona las acciones. Mientras que Q-Learning es un método off-policy, lo que significa que las actualizaciones se realizan utilizando la política óptima conocida, Sarsa es un método on-policy, donde las actualizaciones se realizan utilizando la política que el agente está siguiendo actualmente.

En Q-Learning, la actualización del valor Q para un estado-acción específico se basa en la suposición de que la acción siguiente será la óptima. En contraste, Sarsa actualiza el valor Q basado en la acción real que el agente toma. Esto se refleja en la fórmula de actualización de Sarsa:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\big) \tag{3-6}$$

Esta propuesta permite que el agente aprenda una política más coherente con su comportamiento actual, adaptándose a las condiciones del entorno en tiempo real. Al seguir la política actual para seleccionar acciones futuras, Sarsa garantiza que las decisiones del agente sean consistentes con la política en uso, lo que puede conducir a un comportamiento más estable y predecible en entornos dinámicos.

La principal diferencia con el algoritmo de Q-Learning reside en cómo se calcula el valor esperado de la acción futura en el siguiente estado:

- En Sarsa, la actualización del valor Q se basa en el valor estimado de la próxima acción A_{t+1} en el siguiente estado S_{t+1}, que es la acción que se selecciona según la política actual del agente ($Q(S_t, A_t)$). Esto hace que el aprendizaje esté alineado con la política que se está siguiendo, lo cual es coherente con

el comportamiento real del agente.

- En Q-Learning, la actualización del valor Q se basa en el valor máximo estimado de todas las posibles acciones A_{t+1} en el siguiente estado S_{t+1} ($\max_{A_{t+1}} Q(S_{t+1}, A_{t+1})$). Esto permite al agente aprender la política óptima sin importar la política actual que está siguiendo, enfocándose siempre en el mejor resultado posible.

Este método permite que el agente aprenda una política más coherente con su comportamiento actual, adaptándose a las condiciones del entorno en tiempo real.

3.2.2. Algoritmo de Sarsa

El proceso de actualización de Sarsa implica los siguientes pasos:

1. Inicializar los valores $Q(S_t, A_t)$ arbitrariamente para todos los pares de estado-acción.
2. Repetir para cada episodio:
    a. Inicializar el estado S_0.
    b. Seleccionar una acción A_0 utilizando una política basada en Q.
    c. Para cada paso del episodio:
        i. Tomar la acción A_t, observar la recompensa R_{t+1} y el nuevo estado S_{t+1}.
        ii. Seleccionar una nueva acción A_{t+1} basada en la política.
        iii. Actualizar $Q(S_t, A_t)$ utilizando la fórmula de actualización de Sarsa.
        iv. Establecer $S_t = S_{t+1}$ y $A_t = A_{t+1}$.

3.2.3. Ventajas y desventajas de Sarsa

Sarsa presenta varias ventajas significativas, entre las que destaca la estabilidad. Debido a que Sarsa actualiza los valores Q utilizando la política real que el agente está siguiendo, puede ser más estable en entornos donde las políticas deben adaptarse continuamente. Además, Sarsa tiende a ser menos propenso a la sobreestimación de los valores Q, un problema común en Q-Learning, ya que utiliza las acciones reales en lugar de las óptimas supuestas.

Sin embargo, Sarsa también tiene algunas desventajas. Una de ellas es que puede converger más lentamente que Q-Learning, ya que las acciones exploratorias también influyen en las actualizaciones de los valores Q. Esto puede hacer que el proceso de aprendizaje sea más lento y que se necesite más tiempo para alcanzar una política óptima. Además, la efectividad de Sarsa depende de la política utilizada para seleccionar las acciones, lo que requiere un diseño cuidadoso de dichas políticas.

3.3. Resumen comparativo

3.3.1. Q-Learning

Q-Learning es uno de los algoritmos más básicos y ampliamente utilizados en el aprendizaje por refuerzo. Su simplicidad y eficiencia lo hacen ideal para problemas con un espacio de estado discreto.

Este algoritmo utiliza una tabla de valores Q para almacenar los valores Q para cada par estado-acción, que se actualizan utilizando la ecuación de Bellman, y es particularmente eficiente en entornos donde los estados y las acciones son limitados y bien definidos.

Sin embargo, Q-Learning tiene dificultades cuando se trata de problemas de alta dimensionalidad debido a la explosión del espacio de estado, lo que puede hacer que la tabla Q se vuelva inmanejable.

3.3.2. Sarsa

Similar a Q-Learning, Sarsa también se basa en la actualización de valores Q, pero con una diferencia clave en su enfoque on-policy. Esto significa que Sarsa actualiza la política basada en la acción actual y la siguiente acción tomada por la política actual, lo que puede llevar a una mayor estabilidad en algunos entornos. Esta característica puede ayudar a evitar algunas de las sobreestimaciones que Q-Learning puede experimentar [11].

Sarsa es especialmente adecuado para entornos donde la política debe ser segura y la exploración agresiva debe ser evitada, como en sistemas robóticos sensibles donde las decisiones seguras son cruciales.

4 HERRAMIENTAS UTILIZADAS

La simulación ha sido una parte fundamental en este trabajo sobre aprendizaje por refuerzo, permitiendo probar y validar algoritmos en un entorno controlado. Para lograr esto, se ha hecho uso de una combinación de herramientas que, en conjunto, han creado un ecosistema robusto para la simulación y control de robots.

En primer lugar, se ha trabajado con Gazebo, un simulador de robots que ha proporcionado un entorno realista para probar los comportamientos de los robots en diferentes escenarios. Utilizando Gazebo, ha sido posible implementar modelos de robots, definir sus entornos de operación y ajustar parámetros físicos para asegurar una simulación precisa.

Paralelamente, se ha utilizado ROS como el middleware principal para la comunicación entre los diferentes nodos de la simulación. ROS ha permitido gestionar múltiples robots, facilitando la implementación de algoritmos complejos y la integración de sensores y actuadores. Mediante la creación y modificación de archivos de lanzamiento y scripts en Python, se ha logrado optimizar la interacción entre los robots y su entorno simulado.

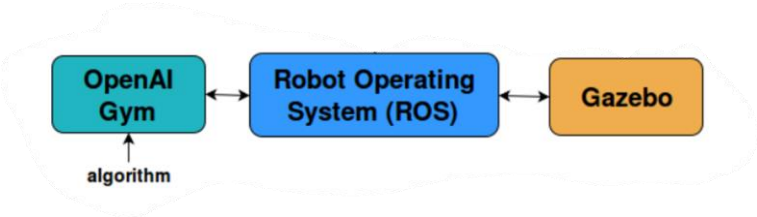


Figura 4-1. Integración OpenAI Gym, ROS y Gazebo para simulación de robots[10]

Finalmente, se ha integrado OpenAI Gym para el desarrollo de algoritmos de aprendizaje por refuerzo, como Q-Learning y Sarsa. Gym ha proporcionado una interfaz estandarizada para la creación de entornos de entrenamiento y la evaluación del rendimiento de los robots en tareas específicas. Adaptar los ejemplos existentes a las necesidades ha sido un proceso desafiante pero gratificante, permitiendo integrar robots que aprenden a navegar de manera eficiente.

[10] https://arxiv.org/pdf/1608.05742

4.1. Gazebo

Gazebo es una plataforma de simulación 3D diseñada para modelar tanto el entorno físico como los robots que operan en él. Se ha utilizado Gazebo como medio de representación gráfica y como la base para las simulaciones debido a la familiarización previa con la herramienta y por sus características avanzadas, a pesar de que pueda ser algo lenta en las simulaciones [15].

4.1.1. Funciones principales

Gazebo es un entorno de simulación realista que ofrece física avanzada, gráficos 3D y sensores simulados, permitiendo replicar con precisión el comportamiento de robots. Utiliza motores físicos de alto rendimiento como ODE, Bullet, Simbody y DART para simular condiciones físicas como gravedad y fricción [16].

Su arquitectura cliente-servidor facilita una simulación distribuida y escalable, con el servidor manejando la simulación y el cliente proporcionando una interfaz gráfica. Integrado estrechamente con ROS, permite una comunicación fluida entre la simulación y los nodos ROS, asegurando que los algoritmos se puedan transferir a robots físicos sin grandes cambios.

La combinación de su realismo en la simulación, su arquitectura distribuida, su integración con ROS, su extensibilidad y el apoyo de una comunidad activa hacen de Gazebo una plataforma ideal para este trabajo en aprendizaje por refuerzo y simulación de robots.

4.2. ROS

ROS (Robot Operating System) es un entorno de trabajo flexible para escribir software de robots. Aunque el nombre podría sugerir que es un sistema operativo completo, ROS es en realidad una colección de herramientas, bibliotecas y convenciones que buscan simplificar la tarea de crear comportamientos complejos y robustos en plataformas robóticas [17].

ROS es altamente modular y flexible, lo que permite su integración con diversas plataformas y hardware. Se ha utilizado ROS Melodic en la distribución de Linux Ubuntu 18.04, que es una de las más comunes para este sistema, y como lenguaje de programación se ha empleado Python, que es el más utilizado junto a C++.

La elección de ROS Melodic se debe a que es la distribución de ROS que se utiliza en los repositorios tomados como referencia, por lo que se ha considerado que mantener esa misma distribución iba a facilitar la integración de las herramientas.

La versión de Python con la que se ha trabajado ha sido Python 2.6, ya que es la que mejor se adapta a los requisitos de las simulaciones y la única que ha permitido ejecutar los programas sin problemas, a pesar de ser una versión actualmente obsoleta. La herramienta OpenAI Gym está especialmente diseñada para ser utilizada con Python 3.7 o versiones superiores, por lo que ha sido necesario hacer algunas modificaciones a la hora de instalar las dependencias para poder usar dicha API sin problemas.

4.2.1. Arquitectura y componentes

Para entender cómo ROS facilita la creación y gestión de comportamientos robóticos complejos, hay que conocer su arquitectura de comunicación y los componentes fundamentales que la componen. La estructura modular de ROS se basa en varios conceptos que permiten la interacción eficiente entre diferentes partes del sistema robótico [17].

- Nodos: Los nodos son los procesos ejecutables en ROS. Cada nodo realiza tareas específicas, y la comunicación entre nodos se realiza a través de mensajes. Esta arquitectura permite distribuir las tareas complejas de un robot en componentes más simples y manejables.
- Mensajes: Los mensajes son estructuras de datos simples que se envían entre nodos. Cada tipo de mensaje tiene un formato específico y se utiliza para transferir datos como sensores o comandos de movimiento, entre otros.

- Temas (tópicos): Los temas son canales de comunicación a los que los nodos pueden publicar o suscribirse para enviar y recibir mensajes. Esto permite una comunicación asíncrona y flexible entre nodos.
- Servicios: Los servicios son otro mecanismo de comunicación en ROS, adecuados para interacciones de solicitud-respuesta. Permiten que un nodo solicite una operación específica y espere una respuesta.
- Acciones: Similar a los servicios, las acciones permiten el manejo de tareas que toman tiempo variable, como mover un brazo robótico a una posición. Permiten el seguimiento del progreso y la interrupción de tareas.

4.2.2. Herramientas

La herramienta ROS cuenta con una serie de herramientas que facilitan la gestión y operación del sistema. A continuación, se presentan las principales herramientas que se han utilizado para gestionar nodos, iniciar configuraciones complejas y visualizar el sistema [17].

- roscore: Es el núcleo de ROS, el cual debe estar ejecutándose para que otros nodos funcionen. Proporciona los servicios básicos de ROS como el registro de nombres y el servidor de parámetros.
- roslaunch: Esta herramienta permite iniciar múltiples nodos y configuraciones de una sola vez a partir de archivos de lanzamiento. Facilita la gestión de configuraciones complejas.
- rqt: Es una interfaz gráfica que proporciona un conjunto de plugins para visualizar y controlar varios aspectos del sistema ROS.
- gazebo_ros: Esta herramienta integra el simulador Gazebo con ROS, permitiendo simular robots en entornos físicos complejos y realistas.

4.3. OpenAI Gym

OpenAI Gym es una herramienta destacada en el campo del aprendizaje por refuerzo, proporcionando una interfaz estándar y una amplia colección de entornos de referencia. Se ha utilizado OpenAI Gym para desarrollar y comparar algoritmos de aprendizaje por refuerzo debido a su simplicidad y flexibilidad [18].

Esta herramienta implementa el ciclo clásico de "agente-entorno" en el que el agente realiza acciones en el entorno y observa cómo cambian los estados del entorno en respuesta a estas acciones. El objetivo en el aprendizaje por refuerzo es manipular el entorno de manera que el agente maximice la recompensa acumulada a lo largo de muchos episodios.

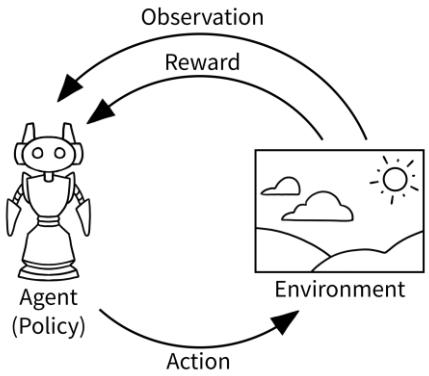


Figura 4-2. Ciclo de interacción en Gym[11]

4.3.1. Gym-Gazebo

Gym-Gazebo es una extensión que integra OpenAI Gym con Gazebo, permitiendo el uso de algoritmos de

[11] https://www.gymlibrary.dev/content/basic_usage/

aprendizaje por refuerzo en entornos de simulación robótica. Esta integración facilita el entrenamiento y evaluación de robots en entornos simulados complejos, combinando la simplicidad de Gym con las capacidades avanzadas de simulación de Gazebo.

4.4. Repositorios

Para desarrollar y simular con estas herramientas, se han tomado como referencia varios repositorios que han proporcionado tanto el código necesario como ejemplos prácticos para llevar a cabo este trabajo.

Se ha utilizado el repositorio de OpenAI Gym como base para entender y aplicar los conceptos de aprendizaje por refuerzo. Este repositorio ha permitido la familiarización con la API estándar de Gym y sus entornos de referencia, facilitando el desarrollo de algoritmos de aprendizaje por refuerzo de manera estructurada [19].

Además, se ha hecho uso del repositorio de Gym-Gazebo para integrar OpenAI Gym con el entorno de simulación Gazebo. Este repositorio proporciona una extensión específica para robots utilizando ROS y Gazebo, permitiendo el uso de técnicas de aprendizaje por refuerzo en entornos de simulación. La documentación y los ejemplos proporcionados en este repositorio han sido fundamentales para configurar y ejecutar simulaciones complejas con robots como el Turtlebot [20].

También se ha consultado el repositorio de ROSbot Description para obtener los modelos y configuraciones necesarios para simular el ROSbot en Gazebo. Este repositorio ha sido especialmente útil para entender cómo modelar este robot específico y cómo integrarlo en el entorno de simulación de Gazebo [21].

Por último, se ha revisado la documentación oficial de Gazebo y los tutoriales de ROS para asegurar que todas las configuraciones y dependencias necesarias estuvieran correctamente implementadas, garantizando así la funcionalidad de las simulaciones.

5 INTEGRACIÓN DE MÚLTIPLES ROBOTS EN SIMULACIÓN GAZEBO

El objetivo principal de este capítulo es documentar de manera detallada el proceso de instalación, configuración y ajuste de un entorno de simulación en Gazebo, utilizando Gym-Gazebo para entrenar tanto a un único robot como a múltiples robots con diferentes algoritmos de aprendizaje por refuerzo. Para ello, fue necesario resolver los problemas técnicos encontrados durante la implementación y proponer soluciones para superarlos.

A continuación, se describe el proceso de instalación y configuración de Gym-Gazebo para la simulación de robots, comenzando con la instalación de los repositorios y dependencias necesarias, así como los pasos de configuración inicial. Se explican los problemas encontrados al integrar el Turtlebot y la decisión de cambiar a ROSbot, detallando las modificaciones realizadas para asegurar su correcto funcionamiento en Gazebo.

También se exploran las mejoras y actualizaciones implementadas en el control del robot con Q-Learning, incluyendo ajustes en las lecturas del LiDAR. El trabajo se extiende a la integración de múltiples robots, adaptando algoritmos de aprendizaje avanzado como Sarsa para entrenar tanto a un único robot como a varios en un entorno de simulación con obstáculos móviles.

5.1. Configuración inicial del entorno de simulación

Para llevar a cabo experimentos con robots en un entorno de simulación controlado, es fundamental configurar correctamente las herramientas y dependencias necesarias.

A continuación, se describen los repositorios y las configuraciones esenciales utilizadas para este propósito, así como los problemas iniciales enfrentados y las soluciones implementadas para asegurar un funcionamiento óptimo del entorno de simulación.

5.1.1 Instalación y configuración de Gym-Gazebo

Para empezar con la configuración del entorno de simulación, se han utilizado dos repositorios principales que proporcionan los scripts y configuraciones necesarios para trabajar con Gym-Gazebo:

1. RoboticsLabURJC Gym-Gazebo. Este repositorio contiene ejemplos y configuraciones específicas para el uso del entorno Gym-Gazebo, pero faltan algunos archivos críticos, lo que hace prácticamente

imposible seguir el tutorial inicial proporcionado [22].

2. Erle Robotics Gym-Gazebo: Este repositorio ha sido modificado para asegurar el correcto funcionamiento del robot en el entorno de simulación. Incluye ejemplos de aplicación de los algoritmos Q-Learning y Sarsa para el entrenamiento del robot Turtlebot en diversos circuitos. En el repositorio se detallan todos los pasos necesarios para la instalación, incluyendo la instalación de todas las dependencias requeridas [23].

Los siguientes pasos dependen de estas dependencias, así como de las herramientas ROS para manejar la comunicación entre los componentes del robot y el entorno de simulación Gazebo.

Después de la configuración inicial, se procedió a configurar el Turtlebot, que es el robot de referencia en muchos de los ejemplos proporcionados. Esta configuración debía asegurar que el Turtlebot pudiera ser simulado correctamente en Gazebo; sin embargo, no fue el caso debido a una serie de problemas que se detallan a continuación.

5.1.2 Problemas iniciales y soluciones

5.1.2.1 Desconexión del tópico de velocidad

Durante la configuración inicial y las pruebas con el Turtlebot en el entorno de simulación de Gym-Gazebo, se encontró un problema crítico: el robot no se movía. Aunque los mensajes de velocidad eran correctos (coincidían con las acciones seleccionadas por el algoritmo de Gym) y se publicaban adecuadamente en el tópico de velocidad, el Turtlebot no respondía a estas instrucciones. Esto se debió a que el tópico */mobile_base/commands/velocity* no estaba conectado a Gazebo, lo cual impedía que el robot Turtlebot ejecutara las acciones seleccionadas. Este tópico es fundamental para enviar comandos de velocidad al robot y permitir su movimiento en el entorno simulado.

5.1.2.2 Cambio del robot de Turtlebot a ROSbot

A pesar de los esfuerzos para solucionar el problema de la desconexión del tópico de velocidad, no se logró que el Turtlebot se moviera en el entorno simulado. Dado que el error provenía de la configuración original del entorno y su modificación resultaba bastante compleja, se optó por cambiar a otro robot con el que ya se contaba con familiaridad: el robot ROSbot de Husarion.

Figura 5-1. ROSbot Husarion[12]

El ROSbot ofrece una arquitectura diferente y, por lo tanto, se esperaba que no presentara los mismos problemas de conexión que el Turtlebot. Para realizar este cambio, se siguieron los siguientes pasos:

[12] https://robosavvy.co.uk/husarion-rosbot-2.html

1. Descargar y configurar ROSbot: Se utilizó el repositorio de Husarion para obtener los archivos de descripción y configuración del ROSbot. Este repositorio incluye los archivos necesarios para definir la estructura del robot, sus sensores y actuadores en el entorno de Gazebo [21].
2. Ajuste de configuraciones: Se modificaron las configuraciones del entorno de simulación para adaptarlas al ROSbot. Esto incluyó cambiar los archivos de lanzamiento y los scripts de control para asegurar el uso de los tópicos y parámetros correctos para el ROSbot.
3. Verificación de comandos de velocidad: Para asegurarme de que los comandos de velocidad se enviaban y recibían correctamente, se utilizó un controlador de velocidad por teclado. Esto permitió confirmar que el ROSbot respondía adecuadamente a los comandos de velocidad en el entorno simulado.
4. Integración en Gym-Gazebo: Finalmente, se integró el ROSbot en el entorno de Gym-Gazebo, reemplazando las configuraciones específicas del Turtlebot por las del ROSbot. Esto implicó modificar los scripts de entrenamiento y los archivos de configuración para asegurar que todos los aspectos de la simulación y el control funcionaran correctamente con el nuevo robot.

El cambio a ROSbot permitió solucionar el problema de comunicación entre el tópico de velocidad y Gazebo. Esta decisión fue clave para superar los problemas iniciales y avanzar en el trabajo con un robot que funcionaba correctamente en el entorno de Gym-Gazebo.

5.2. Configuración de ROSbot en el entorno de simulación

A pesar de que el ROSbot recibía los comandos de velocidad, se presentaban una serie de problemas que impedían la correcta ejecución de la simulación. Al analizar detalladamente todos los ficheros involucrados en la simulación, se lograron solucionar dichos problemas, que se enumeran a continuación.

5.2.1. Actualización del control de velocidad

5.2.1.1. Problemas identificados

Uno de los problemas más críticos en el control del ROSbot fue la inconsistencia en la respuesta del robot a las instrucciones de velocidad durante episodios sucesivos de simulación. Lo que sucedía es que el robot respondía adecuadamente a los comandos de velocidad únicamente en el primer episodio, mientras que en los siguientes el ROSbot repetía de forma indefinida el último movimiento realizado en el primer episodio. Este comportamiento errático sugería que había un problema subyacente en la manera en que se gestionaba el tiempo de simulación y la actualización de las variables de control.

Al analizar los mensajes publicados en el tópico */cmd_vel*, se observó que las velocidades transmitidas coincidían con las acciones seleccionadas por el algoritmo de aprendizaje. No obstante, la falta de respuesta correcta del robot en los episodios posteriores llevó a investigar más profundamente la comunicación entre este tópico y el entorno de simulación de Gazebo. Se descubrió que el problema estaba en la actualización del tiempo de simulación, específicamente en cómo se manejaban los valores de tiempo cuando la simulación se reiniciaba después de cada episodio.

5.2.1.2. Soluciones implementadas

Para resolver el problema del comportamiento anómalo del ROSbot en la simulación de Gazebo, se trabajó en el plugin *gazebo_ros_skid_steer_drive*, encargado de la tracción diferencial del robot y la publicación de datos de odometría. Se identificó que una de las funciones de este plugin, esencial para actualizar las velocidades del robot, no manejaba adecuadamente los reinicios de la simulación, lo que causaba fallos en la actualización de las velocidades de las ruedas.

Para solucionar esto, se añadió una verificación en dicha función para detectar cuándo el tiempo transcurrido desde la última actualización era negativo, señal de un reinicio de la simulación. Al detectar esta condición, se restableció el tiempo de la última actualización al tiempo actual de la simulación, asegurando que las velocidades se actualizaran correctamente tras cada reinicio.

Este ajuste garantizó que el controlador de velocidad funcionara adecuadamente después de reiniciar la simulación, permitiendo un comportamiento consistente del ROSbot. Además, se realizó un análisis detallado del comportamiento del robot en la simulación, imprimiendo variables para detectar anomalías. Estos esfuerzos mejoraron considerablemente la estabilidad del ROSbot en el entorno de simulación de Gazebo, optimizando su capacidad respuesta a los comandos de velocidad.

5.2.2. Ajuste en las lecturas del LiDAR

5.2.2.1. Comparación entre Turtlebot y ROSbot

Durante la configuración y optimización del ROSbot en el entorno de simulación Gym-Gazebo, se descubrió que existían diferencias significativas en las lecturas del LiDAR al comparar los datos obtenidos del Turtlebot y el ROSbot. Estas diferencias no solo afectaban la densidad de las lecturas, sino también el rango de detección y la disposición de los sensores.

El LiDAR del Turtlebot generaba un menor número de lecturas y tenía un rango de detección más limitado comparado con el del ROSbot, ya que mientras que el Turtlebot toma alrededor de 20 medidas en un barrido de 270 grados, el ROSbot es mucho más detallado, con 720 medidas a lo largo de un barrido completo de 360 grados. Esta densidad de lecturas proporcionaba al ROSbot una visión mucho más detallada de su entorno. Además, el punto de inicio y finalización del barrido del LiDAR también difería entre ambos robots.

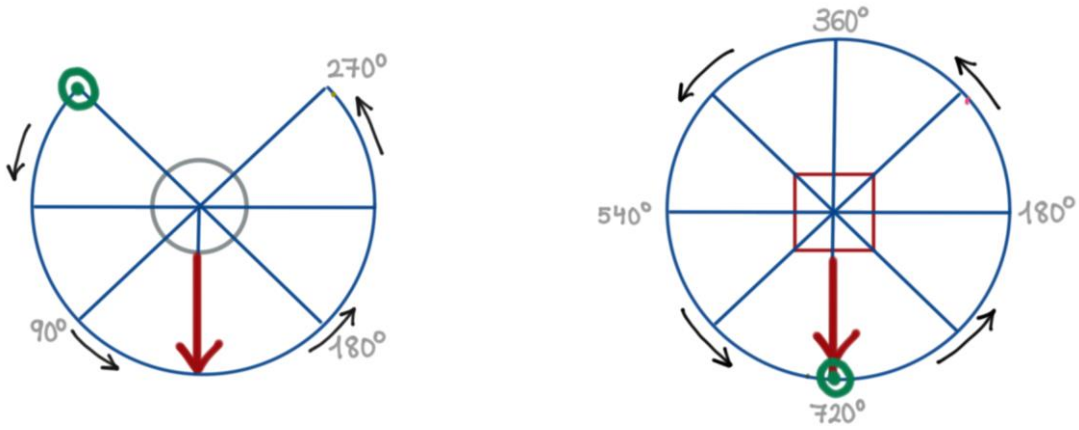


Figura 5-2. Medidas iniciales de LiDAR de Turtlebot y ROSbot

En la figura 5-2 se muestran las medidas iniciales de LiDAR que acaban de ser descritas, tanto del Turtlebot (diagrama de la izquierda) como del ROSbot (diagrama de la derecha). En estos diagramas, la flecha roja marca la parte frontal del robot, mientras que el punto verde y las flechas negras indican el punto de comienzo de las lecturas y el sentido en el que son tomadas, respectivamente.

Aunque esta diferencia no era esencial ni relevante para el aprendizaje, se consideró importante realizar una modificación para asemejar ambas lecturas, dado que se estaba en el proceso de comprender y aprender los algoritmos. Se buscó que el ROSbot funcionara de una manera lo más similar posible al Turtlebot para facilitar la transición y mantener la analogía en los experimentos y análisis.

5.2.2.2. Modificaciones para asemejar lecturas

Para solucionar las discrepancias en las lecturas del LiDAR entre el Turtlebot y el ROSbot, se realizaron ajustes en los scripts de procesamiento de datos del sensor. El objetivo fue adaptar las lecturas del ROSbot para que se asemejaran lo máximo posible a las del Turtlebot, logrando así una coherencia en las observaciones utilizadas por el algoritmo de aprendizaje.

Primero, se redujo el número de lecturas del LiDAR del ROSbot de 720 a 540 para cubrir el mismo rango que cubrían las 20 lecturas del Turtlebot. Se ajustó el rango del ROSbot para que solo cubriera los 270 grados

necesarios, alineándose así con el rango de detección del Turtlebot.

Finalmente, se adaptó el procesamiento de estas lecturas en el algoritmo de aprendizaje. Se modificó cómo el algoritmo discretizaba las observaciones del LiDAR para utilizar el mismo número de lecturas y en el mismo orden que las del Turtlebot. Esta modificación aseguró que el algoritmo de aprendizaje tuviera una visión uniforme del entorno, independientemente del robot utilizado, permitiendo así una transferencia directa de las estrategias de aprendizaje entre ambos robots.

Una vez solucionados estos problemas en la simulación, se pudieron llevar a cabo los entrenamientos del ROSbot con el algoritmo Q-Learning de aprendizaje por refuerzo.

5.2.3. Arquitectura del sistema

A continuación, se muestra un análisis de la organización de tópicos y nodos en el entrenamiento de aprendizaje por refuerzo del ROSbot. Este esquema es esencial para comprender la gestión de las comunicaciones dentro del sistema.

Además, el diagrama ilustra las interacciones entre los distintos componentes, permitiendo visualizar cómo los datos se intercambian y se procesan entre los diferentes módulos del sistema, lo que facilita la identificación de posibles problemas de comunicación y la optimización del rendimiento general del robot.

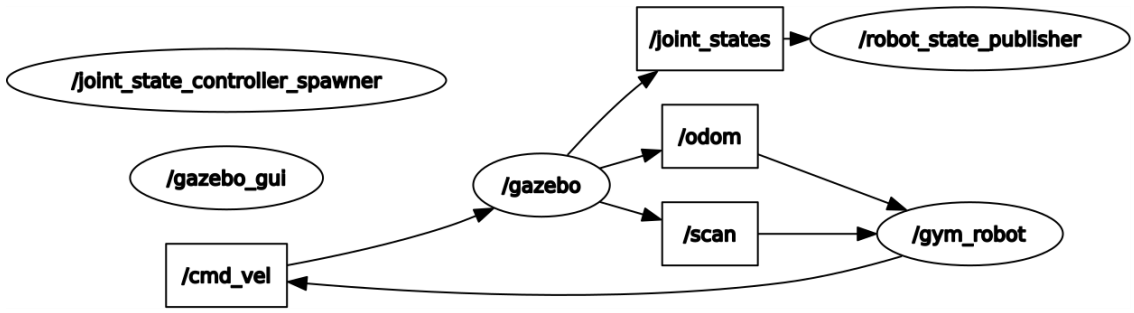


Figura 5-3. Diagrama de tópicos y nodos en ROS para la simulación en Gazebo para un único robot

La figura 5-3 muestra el diagrama de tópicos y nodos en el entrenamiento del ROSbot, siendo el nodo principal identificado como */gym_robot*. El nodo */gym_robot* interactúa con varios tópicos y nodos relacionados con la simulación en Gazebo.

El nodo */gym_robot* lee de los tópicos */odom* y */scan* y publica en el tópico */cmd_vel*:

- */odom*: Recibe datos de odometría para la localización y movimiento del robot.
- */scan*: Tópico que recoge datos del escáner láser (LiDAR).
- */cmd_vel*: Envía comandos de velocidad al robot.

Aparte de estos tópicos, en la simulación existen otros que colaboran en la ejecución de Gazebo, como */joint_states*, que recibe información del estado de las articulaciones del robot. Estos datos se utilizan para actualizar el estado general del robot en el tópico */robot_state_publisher*, que es consumido por otros nodos para diversas funciones, incluyendo la visualización y la planificación de movimientos.

Los nodos */gazebo* y */gazebo_gui* son fundamentales para la simulación, ya que gestionan la interacción de los robots con el entorno simulado en Gazebo. Estos nodos están conectados a todos los tópicos esenciales de ambos robots, permitiendo una simulación precisa y en tiempo real. La interfaz gráfica de usuario de Gazebo *(/gazebo_gui*) facilita la visualización de la simulación, lo que es crucial para el monitoreo y la evaluación del desempeño de los robots durante el entrenamiento.

Este diagrama es importante para visualizar cómo el nodo */gym_robot* integra la información de diferentes sensores y fuentes dentro de un entorno de simulación en Gazebo para controlar un robot. Además, muestra la interconexión con otros nodos y tópicos, permitiendo una comprensión clara del flujo de datos dentro del sistema.

5.3. Integración de múltiples robots

La integración de múltiples robots en entornos de simulación requiere una configuración precisa para asegurar un funcionamiento eficiente y sincronizado.

En este punto, se abordan los siguientes problemas en la integración de múltiples robots en la simulación: la configuración y ejecución simultánea de dos robots mediante archivos de lanzamiento, modificaciones en los scripts de control para manejar comandos y datos de odometría sin conflictos, la solución para el reinicio completo de la simulación con una alternativa de reinicio parcial y ajustes en el espacio de observaciones para incluir la distancia entre robots.

5.3.1. Configuración y ejecución de dos robots

5.3.1.1. Archivos de lanzamiento múltiple

Para iniciar dos robots simultáneamente en el entorno de simulación, se creó un archivo de lanzamiento. Este archivo resulta esencial para configurar y gestionar múltiples instancias del ROSbot en Gazebo, permitiendo definir las instancias de los robots, sus posiciones iniciales y los parámetros necesarios para asegurar que cada robot se comporte de manera independiente y coordinada.

El archivo de lanzamiento configura cada ROSbot con sus propios tópicos de control y sensores, permitiendo que ambos robots operen sin interferir uno con el otro. Esto incluye la asignación de diferentes nombres de robots y el remapeo de tópicos específicos para cada uno, como los de velocidad y escaneo LiDAR.

Además, el archivo de lanzamiento incluye la configuración de los servicios necesarios para gestionar la simulación, como la inserción de modelos y la inicialización de los controladores de estado de los robots. Esto asegura que cada instancia del ROSbot esté correctamente configurada y lista para el entrenamiento y la simulación.

5.3.1.2. Modificaciones en los scripts de control

Para manejar múltiples robots de manera efectiva, se realizaron varias modificaciones en los scripts de control. Estos ajustes garantizaron que cada robot tuviera asignaciones de tópicos de velocidad y odometría independientes, evitando así cualquier conflicto de comunicación entre los robots.

Se implementaron parámetros específicos para cada robot, tales como el nombre del robot y su posición inicial, los cuales se pasan al script desde el archivo de lanzamiento.

Los scripts fueron adaptados para remapear los tópicos de velocidad y odometría a tópicos únicos para cada robot, permitiendo que cada robot responda solo a los comandos dirigidos a él y no a los de otros robots en la simulación.

También se modificaron los scripts de control para reiniciar las posiciones de los robots de manera independiente al finalizar cada episodio de entrenamiento. Esto incluye la actualización de las variables de entrenamiento sin reiniciar la simulación completa, facilitando un aprendizaje continuo y más eficiente.

Se añadieron funciones específicas para gestionar la posición inicial y el estado del otro robot, suscribiéndose a los tópicos de odometría de ambos robots, asegurando que cada robot pueda acceder a la información de la posición del otro en tiempo real.

5.3.2. Reinicio independiente de la simulación

5.3.2.1. Problemas con el reinicio completo

En el desarrollo del trabajo, se encontró un problema relacionado con el reinicio completo de la simulación al trabajar con más de un robot. Originalmente, cada vez que un episodio de entrenamiento concluía, el script estaba programado para reiniciar toda la simulación. Esto, sin embargo, tenía un impacto negativo en el proceso de aprendizaje para múltiples robots.

El problema principal era que, al reiniciar toda la simulación, no se permitía que el segundo robot completara su

episodio de aprendizaje. El primer robot, al finalizar su episodio, forzaba un reinicio del entorno completo, lo que interrumpía el episodio en curso del segundo robot. Esto resultaba en que el segundo robot no podía seguir acumulando experiencia de manera continua, afectando negativamente a la eficiencia del entrenamiento combinado de ambos robots.

Este comportamiento no solo reducía la efectividad del aprendizaje, sino que también incrementaba el tiempo total requerido para entrenar a los robots, ya que muchos episodios se veían truncados y no contribuían de manera completa al proceso de aprendizaje.

5.3.2.2. Solución implementada para el reinicio parcial

Para solucionar este problema, se implementó una modificación en el script que permite un reinicio parcial de la simulación, centrado específicamente en reiniciar solo las posiciones y las variables de entrenamiento de los robots, sin reiniciar el tiempo de simulación ni el entorno completo.

Esta solución se basa en varias estrategias:

1. Reinicio de posiciones: El script fue configurado para que, al final de cada episodio, los robots regresaran a sus posiciones iniciales predefinidas. Esto permite que los robots comiencen cada nuevo episodio desde un punto de partida conocido, facilitando el análisis y la comparación de resultados entre episodios.
2. Reinicio de variables de entrenamiento: Además de las posiciones, se reinician las variables internas de cada robot relacionadas con el proceso de aprendizaje. Esto incluye reiniciar las métricas de desempeño y las variables de estado utilizadas por los algoritmos de aprendizaje, asegurando que cada nuevo episodio se inicie con un estado limpio y sin influencias residuales del episodio anterior.
3. Mantenimiento del tiempo de simulación: En este nuevo caso, el tiempo de simulación no se reinicia. Al mantener el tiempo de simulación continuo, se evita el problema de las inconsistencias temporales que pueden surgir al reiniciar completamente el entorno.
4. Independencia de reinicio entre robots: Cada robot puede completar su episodio y reiniciar de manera independiente del otro. Esto significa que, si un robot choca y termina su episodio, solo ese robot se reinicia y comienza un nuevo episodio, mientras que el otro robot continúa con su episodio actual sin interrupciones. Esta independencia es esencial para permitir un aprendizaje eficiente y sin interrupciones para ambos robots.

Estas modificaciones resultaron en una mejora significativa en el proceso de entrenamiento, permitiendo un aprendizaje más continuo y eficiente para ambos robots, optimizando el uso del tiempo de simulación y mejorando la calidad del entrenamiento.

5.3.3. Ajustes en el espacio de observaciones

5.3.3.1. Inclusión de la distancia entre robots

Para mejorar la navegación y evitar colisiones entre múltiples ROSbot, se añadió la distancia relativa entre ellos al espacio de observaciones. Esto permite que, además de las lecturas del LiDAR, cada robot tenga información sobre la posición del otro robot en relación con la suya, proporcionando un contexto más completo del entorno inmediato y facilitando decisiones más informadas durante el desplazamiento.

La distancia entre robots se mide en términos de las coordenadas X e Y, proporcionando una representación espacial precisa que ayuda a prever y evitar posibles colisiones.

5.3.3.2. Modificaciones en los scripts de observación

Se actualizaron los scripts de observación para suscribirse a los tópicos de odometría de ambos robots, obteniendo la posición exacta de cada uno. La posición del robot propio se obtiene del tópico */odom* correspondiente, mientras que la posición del otro robot se remapea desde el archivo de lanzamiento para ser accesible desde el mismo script.

Los scripts de observación fueron ajustados para normalizar estas posiciones, asegurando que las distancias entre

robots sean interpretadas uniformemente dentro del rango esperado por el algoritmo de aprendizaje, facilitando decisiones sólidas y basadas en datos precisos.

5.3.4. Arquitectura del sistema

Para entender la complejidad y la organización del entorno de simulación multi-robot en ROS, es fundamental analizar la interconexión de los diversos nodos y tópicos que permiten la operación sincronizada y eficiente de los robots. Los elementos esenciales que permiten el funcionamiento de los dos robots se centran en varios tópicos y nodos que gestionan y transmiten datos críticos para la operación y la simulación en tiempo real.

A continuación, se detallan los componentes y las conexiones que facilitan la comunicación entre los robots y el entorno de simulación.

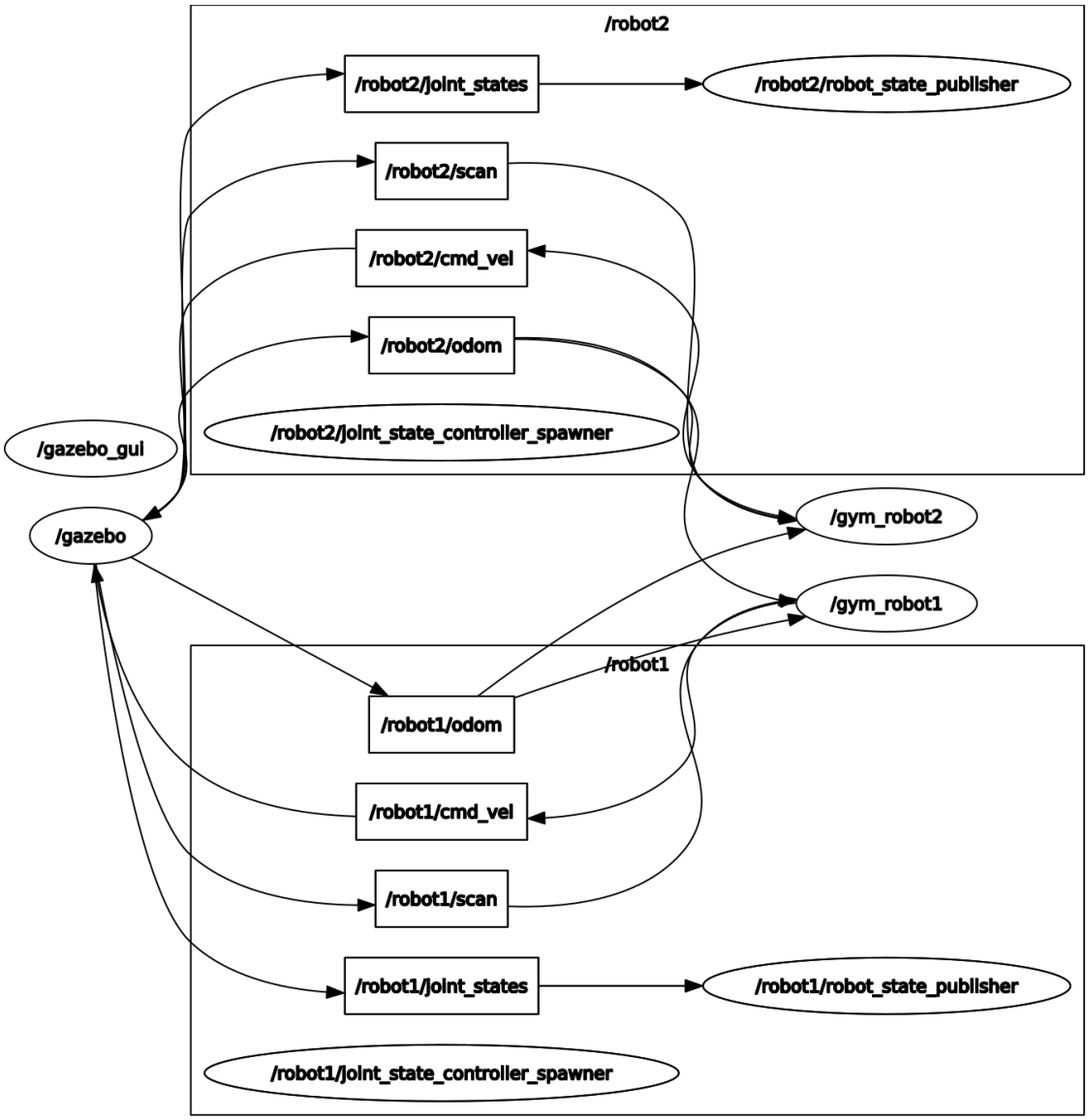


Figura 5-4. Diagrama de tópicos y nodos en ROS para la simulación en Gazebo para sistema multi-robot

En la figura 5-4 se muestra la arquitectura del sistema ROS para el entorno de entrenamiento multi-robot. Los elementos esenciales que permiten el funcionamiento de los dos robots (*/robot1* y */robot2*) se centran en varios tópicos y nodos que gestionan y transmiten datos críticos para la operación y la simulación en tiempo real.

Los dos robots en el sistema tienen sus propios conjuntos de tópicos y nodos que gestionan su operación básica. Para cada robot, hay tópicos dedicados a la odometría *(/robot1/odom* y */robot2/odom*), que proporcionan

información sobre la posición y el movimiento del robot. Los comandos de velocidad se manejan a través de los tópicos */robot1/cmd_vel* y */robot2/cmd_vel*, permitiendo el control del movimiento de cada robot. Además, los datos del escáner láser, que son cruciales para la navegación y la detección de obstáculos, se transmiten a través de */robot1/scan* y */robot2/scan*.

La comunicación entre los robots y el entorno de simulación se gestiona a través de varios nodos y servicios. Los nodos */gym_robot1* y */gym_robot2* actúan como intermediarios entre los datos específicos de cada robot y el entorno de entrenamiento en gym, suscribiéndose a los tópicos de odometría, comandos de velocidad, escáner láser y estados de las articulaciones de cada robot. Esto permite que el entorno de entrenamiento en gym reciba y envíe información relevante para la simulación y el aprendizaje.

El sistema está diseñado de manera que cada robot opera de manera autónoma pero coordinada dentro del entorno de simulación y entrenamiento. Los datos esenciales de cada robot se transmiten a través de tópicos específicos, permitiendo la actualización continua del estado y el control preciso de los movimientos. La comunicación entre los robots y el entorno de simulación se facilita a través de nodos intermediarios y servicios que aseguran que toda la información relevante esté disponible para el entrenamiento y la simulación en tiempo real.

5.4. Extensión a algoritmos de aprendizaje avanzado

En esta etapa, el objetivo fue extender las capacidades de aprendizaje del ROSbot utilizando un algoritmo avanzado como Sarsa. La principal motivación detrás de esta extensión fue mejorar el desempeño del robot en tareas complejas mediante técnicas de aprendizaje reforzado más sofisticadas.

5.4.1. Sarsa

5.4.1.1. Adaptación de los códigos originales

Para implementar el algoritmo de Sarsa en el ROSbot, fue necesario adaptar los códigos originales diseñados inicialmente para un solo robot. Estos algoritmos estaban destinados a entrenar al robot en entornos simulados utilizando técnicas de aprendizaje por refuerzo, permitiendo al robot aprender a tomar decisiones óptimas basadas en sus interacciones con el entorno.

Durante el proceso de adaptación, se modificaron las estructuras de datos y las funciones de observación de manera similar a las modificaciones realizadas para el algoritmo inicial. Dado que el algoritmo Sarsa funciona de manera prácticamente idéntica al Q-Learning a nivel de código, los cambios realizados fueron los mismos.

Además, se modificó el manejo de los tópicos de comunicación entre los robots y el entorno de simulación. Los tópicos de comandos de velocidad y odometría fueron remapeados para cada robot, permitiendo que los algoritmos de aprendizaje enviaran y recibieran información específica de cada ROSbot sin interferencias.

5.4.1.2. Configuración para un robot y múltiples robots

Para facilitar el uso de los algoritmos de aprendizaje avanzados con uno o varios ROSbot, se crearon archivos de lanzamiento específicos. Estos archivos permiten configurar y ejecutar los entrenamientos de manera sencilla y eficiente.

Para entrenamientos con un solo robot, el archivo de lanzamiento configura el entorno de simulación para que solo un ROSbot esté activo. Esto incluye la inicialización de los parámetros de entrenamiento, la configuración del entorno de Gazebo y la definición de los tópicos de comunicación necesarios.

Para entrenamientos con múltiples robots, el archivo de lanzamiento se encarga de inicializar varios ROSbot en el entorno de simulación. Cada robot se lanza con su propio conjunto de parámetros de entrenamiento y se asegura de que las observaciones y comandos sean específicos para cada uno. Esto implica una gestión cuidadosa de los tópicos de comunicación y la sincronización de los estados de los robots para evitar conflictos.

Estos archivos de lanzamiento permiten una gran flexibilidad en los experimentos de aprendizaje, facilitando la transición entre escenarios de entrenamiento de un solo robot a entornos multi-robot, y asegurando que el algoritmo de Sarsa pueda aplicarse eficazmente en ambos contextos.

6 ENTRENAMIENTO Y RESULTADOS

En este capítulo se presentan los resultados del entrenamiento de robots tanto individuales como con obstáculos móviles utilizando distintos algoritmos de aprendizaje por refuerzo. El propósito principal es evaluar y comparar la eficacia de cada algoritmo dentro de un mismo escenario, así como analizar cómo diferentes parámetros dentro de cada algoritmo influyen en el rendimiento del aprendizaje y el comportamiento de los robots.

El objetivo general del entrenamiento es optimizar el comportamiento de los robots para que puedan realizar tareas específicas de manera eficiente y autónoma. Para ello, se pretende entender cómo distintos algoritmos de aprendizaje por refuerzo, así como los parámetros que los regulan, afectan al rendimiento de los robots, tanto cuando operan de manera individual como multi-robot.

Para analizar con más precisión la evolución de las recompensas obtenidas por el robot a lo largo de los episodios, se utilizarán medias móviles. Las medias móviles son una herramienta estadística que permite suavizar las fluctuaciones a corto plazo y destacar las tendencias a largo plazo en los datos. En el contexto del aprendizaje por refuerzo, donde las recompensas pueden variar significativamente de un episodio a otro debido a la exploración y explotación, las medias móviles proporcionan una visión más clara y comprensible del rendimiento general del agente.

En concreto, se ha elegido la media móvil de 100 episodios, ya que permite suavizar adecuadamente las fluctuaciones a corto plazo y capturar tendencias a largo plazo en el rendimiento del robot, proporcionando un balance óptimo entre detalle y claridad. Esto facilita la identificación de períodos de mejora continua, estabilización o retrocesos en el aprendizaje, y permite evaluar de manera efectiva el impacto de los parámetros del algoritmo en el rendimiento del robot.

6.1. Entorno de entrenamiento

El entorno donde se realizan todos los entrenamientos es un circuito cerrado configurado con paredes que le dan la apariencia de un laberinto. Cada vez que el robot choca con una pared, recibe una penalización y es reiniciado desde la posición inicial. Cuando el robot completa el recorrido exitosamente y regresa al punto de partida, comienza un nuevo episodio. Este diseño de entorno desafía al robot a aprender a evitar obstáculos y a planificar rutas eficientes para maximizar las recompensas y minimizar las penalizaciones.

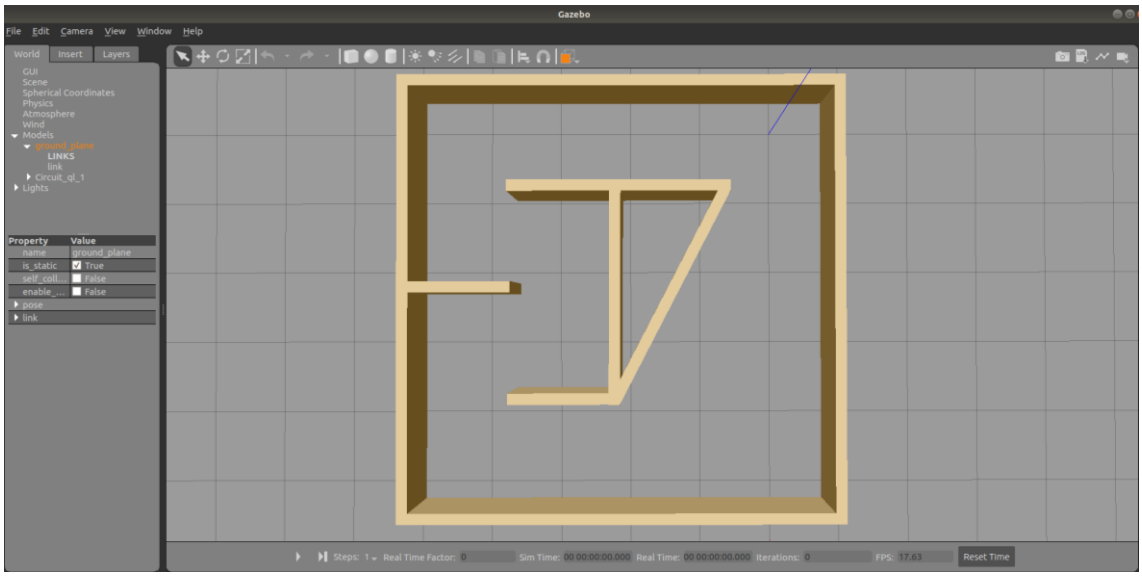


Figura 6-1. Entorno de simulación en los entrenamientos de un único robot

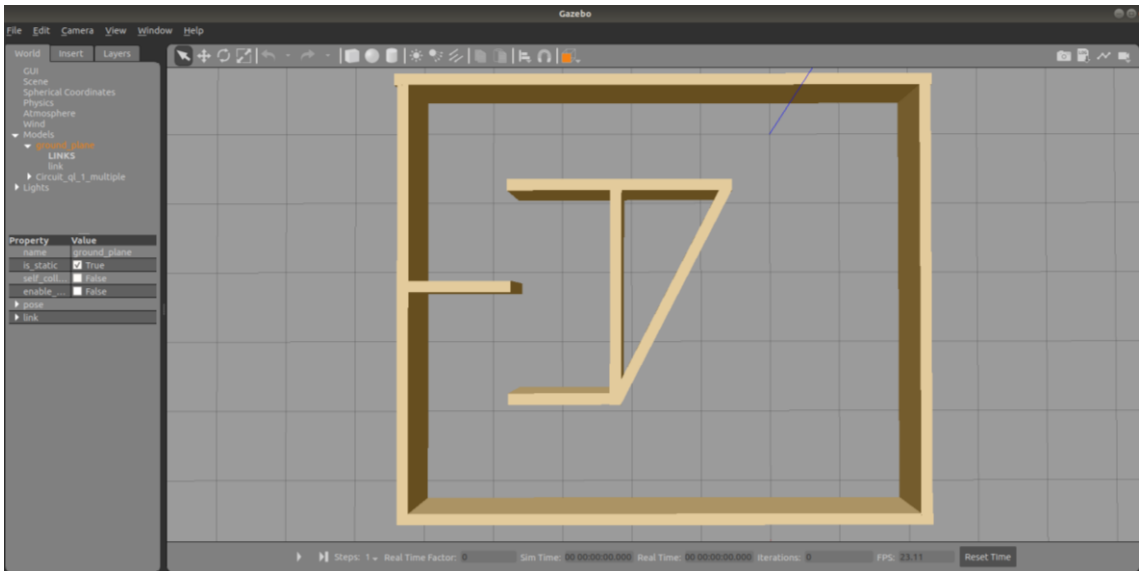


Figura 6-2. Entorno de simulación en los entrenamientos multi-robot

Como se muestra en las figuras 6-1 y 6-2, el entorno de simulación destinado al caso de múltiples robots presenta un lateral más ancho en comparación con el entorno para un único robot. Esta diferencia en el diseño se debe a que, en el escenario de simulación con varios robots, ambos inician sus trayectorias desde posiciones relativamente cercanas entre sí.

Por lo tanto, es fundamental que el entorno ofrezca un espacio suficiente para que los robots puedan comenzar la simulación desde puntos casi idénticos, en lugar de arrancar desde ubicaciones diferentes dentro del recorrido. Este espacio adicional es crucial para garantizar que cada robot tenga las mismas condiciones iniciales, lo cual facilita una comparación más justa y precisa de su rendimiento durante el entrenamiento, además de evitar posibles interferencias o colisiones que podrían surgir si los robots estuvieran demasiado aglomerados al iniciar.

6.2. Entrenamiento de un solo robot

El entrenamiento de un solo robot se llevará a cabo utilizando dos algoritmos diferentes: Q-Learning y Sarsa. La finalidad de este análisis es identificar las fortalezas y debilidades de cada algoritmo en un contexto aislado, permitiendo determinar cuál de ellos es más adecuado para tareas individuales y en qué circunstancias. Al centrarnos en un robot individual, se podrá evaluar la capacidad de cada algoritmo para optimizar el comportamiento de un agente autónomo en un entorno controlado.

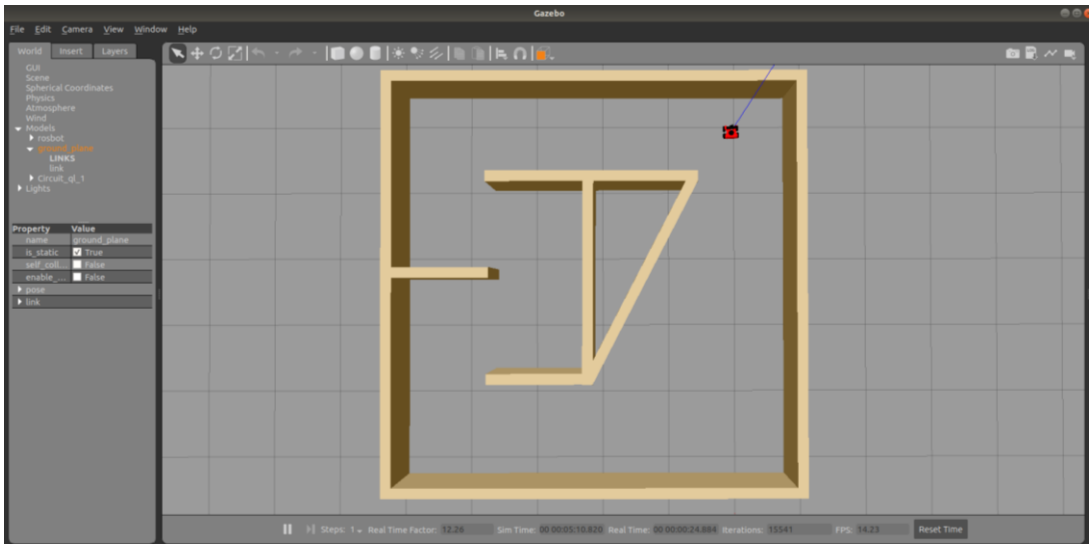


Figura 6-3. Entorno de simulación de un único robot

6.2.1. Q-Learning

6.2.1.1. Entrenamiento 1

En el primer entrenamiento se utiliza una configuración de parámetros que según la teoría es la más adecuada y la que maximiza el rendimiento del aprendizaje por refuerzo. Por ello, en la primera simulación realizada para entrenar a un robot individual utilizando el algoritmo Q-Learning, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,1	0,8	0,9	0,999

Tabla 6-1. Parámetros del algoritmo Q-Learning en el primer entrenamiento

Alfa, también conocido como la tasa de aprendizaje, determina cuánto influye el nuevo valor de Q en la actualización de los valores Q existentes. Un valor de alfa de 0,1 significa que el nuevo valor Q tiene un impacto del 10% en la actualización, mientras que el valor existente tiene un impacto del 90%. Este parámetro es crítico para equilibrar entre el aprendizaje rápido y la estabilidad de los valores aprendidos.

Gamma, conocido como el factor de descuento, define la importancia de las recompensas futuras. Un valor de gamma de 0,8 indica que se valoran las recompensas futuras, pero con un descuento del 20% en cada paso. Este parámetro ayuda a enfocar el aprendizaje en acciones que no solo son inmediatamente beneficiosas, sino que también proporcionan recompensas a largo plazo.

Épsilon es el parámetro de exploración-explotación que determina la probabilidad de que el robot elija una acción aleatoria (exploración) en lugar de la mejor acción conocida (explotación). Un valor inicial de épsilon de 0,9 significa que, al comienzo del entrenamiento, hay un 90% de probabilidad de que el robot explore acciones nuevas. Este alto valor inicial fomenta la exploración al inicio del aprendizaje.

El descuento de épsilon define la tasa a la cual épsilon disminuye después de cada episodio. Un descuento de 0,999 significa que épsilon se multiplica por 0,999 después de cada episodio, reduciendo gradualmente la probabilidad de exploración y aumentando la probabilidad de explotación de las acciones conocidas. Este decremento progresivo permite que el robot explore al principio y, a medida que aprende, se centre más en explotar las estrategias más exitosas.

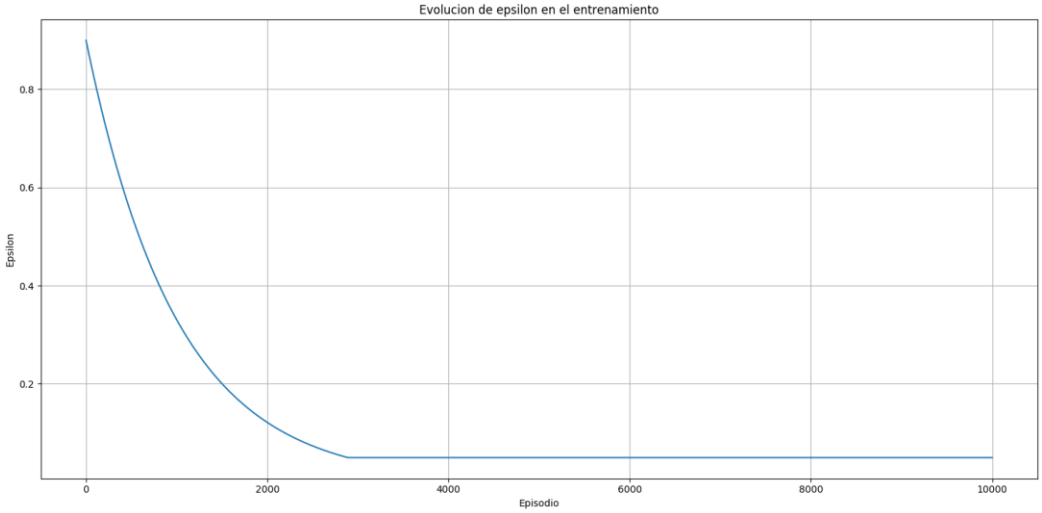


Figura 6-4. Evolución de épsilon a lo largo del entrenamiento con un decrecimiento de épsilon de 0,999

En la figura 6-4 se puede observar el valor de épsilon a lo largo de los episodios que componen el entrenamiento. Épsilon decrece rápidamente hasta aproximadamente el episodio 3000, momento en el cual el agente pasa a centrarse en explotar las políticas exitosas aprendidas en los episodios previos, siendo mínimas las ocasiones en las que se exploran nuevas acciones. Esta estrategia permite al agente consolidar sus conocimientos y maximizar las recompensas obtenidas, aunque puede llevar a una menor capacidad de adaptación a nuevas situaciones debido a la falta de exploración en etapas más avanzadas del entrenamiento.

Las trayectorias seguidas por el robot en cada uno de los episodios se muestran a continuación, con el objetivo de poder visualizar el comportamiento del robot antes de pasar a analizar las recompensas obtenidas.

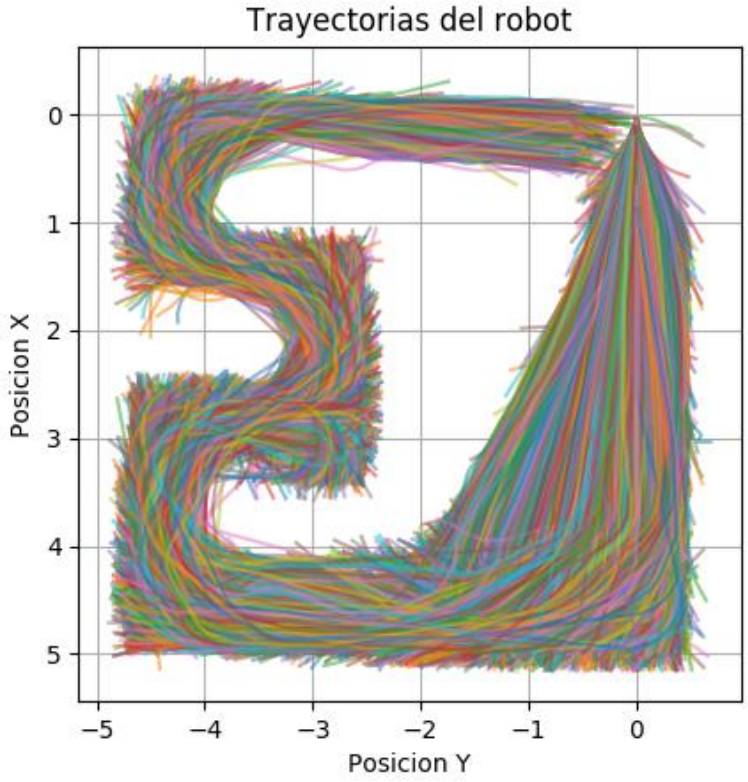


Figura 6-5. Trayectoria realizada por el robot en cada uno de los episodios del entrenamiento

La figura 6-5 muestra un conjunto de trayectorias seguidas por un robot, todas comenzando desde el punto de origen en las coordenadas. Este punto inicial, ubicado en la parte derecha del gráfico, es donde el robot inicia su desplazamiento por el plano de coordenadas X-Y, explorando diversas rutas posibles. Estas trayectorias son el resultado del aprendizaje del robot a través del proceso de optimización dominado por el algoritmo Q-Learning.

El algoritmo Q-Learning es responsable de guiar al robot en su exploración del entorno, permitiéndole aprender qué acciones tomar en cada caso. A medida que el robot se aleja del origen, las trayectorias muestran una dispersión notable, lo que sugiere que el robot está evaluando y aprendiendo diferentes rutas posibles. Cada línea en la figura representa una posible trayectoria que el robot podría seguir, indicando la variabilidad en su movimiento, que es característica del proceso de aprendizaje adaptativo propio del Q-Learning.

Por tanto, esta figura visualiza cómo el algoritmo Q-Learning ha dominado el proceso de aprendizaje del robot, guiándolo a través de un proceso iterativo de prueba y error para encontrar trayectorias óptimas en el espacio desde un punto de partida común. La diversidad y la distribución de las trayectorias reflejan el aprendizaje adaptativo y la capacidad del robot para explorar y explotar diferentes rutas bajo la influencia de este potente algoritmo de aprendizaje por refuerzo.

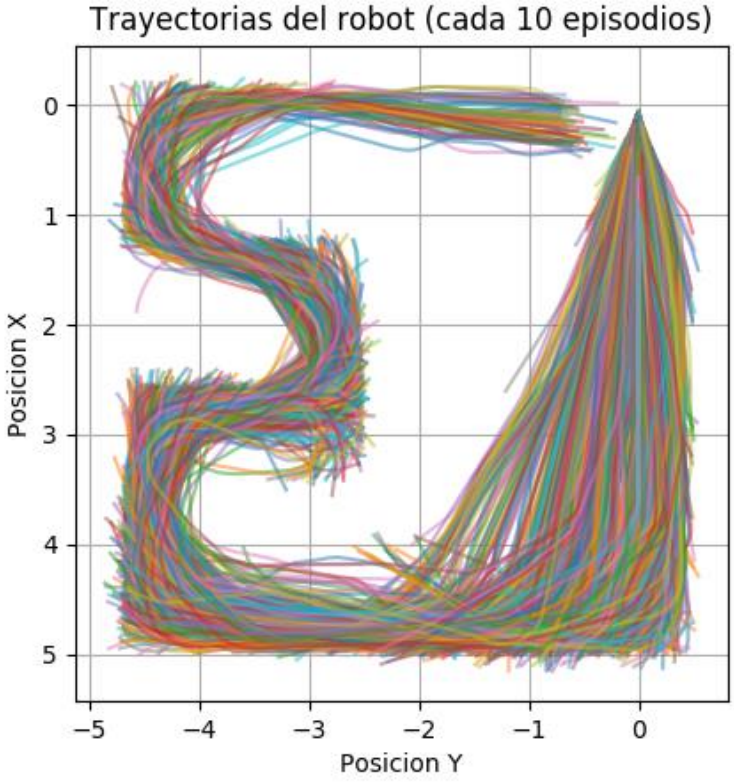


Figura 6-6. Trayectoria realizada por el robot en uno de cada diez episodios del entrenamiento con Q-Learning

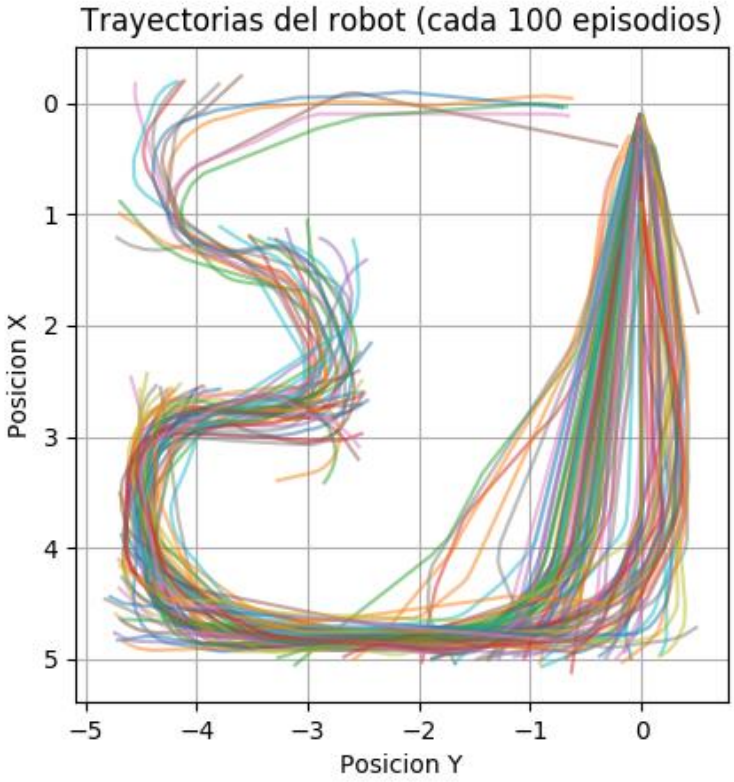


Figura 6-7. Trayectoria realizada por el robot en uno de cada cien episodios del entrenamiento con Q-Learning

Se han incluido estas dos figuras adicionales para ofrecer una perspectiva más completa sobre el progreso del aprendizaje del robot a través del tiempo. La figura con trayectorias de un episodio de cada diez permite observar detalles y ajustes en las rutas del robot, mostrando cómo refina su estrategia a medida que explora el entorno. En contraste, la figura con trayectorias de un episodio de cada cien destaca las trayectorias más consolidadas y estables, reflejando la optimización y especialización en su política de navegación tras una fase prolongada de aprendizaje. Juntas, estas representaciones muestran tanto el proceso de ajuste continuo como la consolidación de comportamientos óptimos del robot.

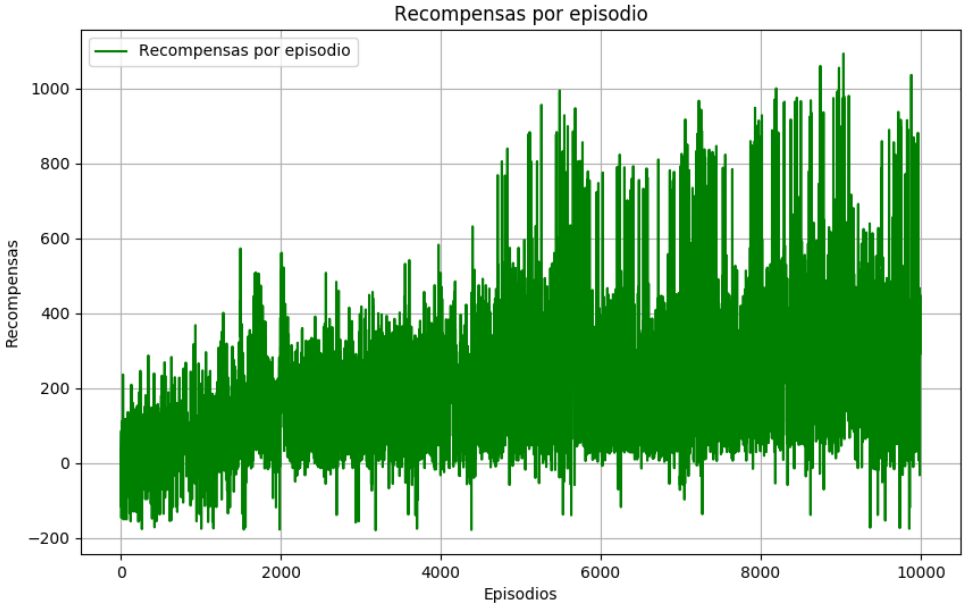


Figura 6-8. Recompensas por episodio en el primer entrenamiento con Q-Learning

La figura 6-8 muestra las recompensas obtenidas por el robot en cada uno de los 10.000 episodios de entrenamiento. Se puede observar un patrón de alta variabilidad en las recompensas individuales; sin embargo, al mismo tiempo, se puede apreciar un incremento general en la tendencia de las recompensas a lo largo del tiempo. Este incremento en las recompensas promedio es una señal positiva, ya que indica que el robot está aprendiendo a tomar decisiones más efectivas que le permiten obtener mejores resultados en el entorno en el que está operando. Esta progresión es esperada en algoritmos de aprendizaje como el Q-Learning, donde el agente aprende a maximizar sus recompensas acumuladas a lo largo de múltiples episodios.

Por otro lado, la alta variabilidad observada en las recompensas también es un aspecto importante que considerar, ya que la gran dispersión de puntos en el gráfico indica que en algunos episodios las recompensas fueron significativamente altas, mientras que en otros fueron muy bajas. Esta variabilidad es común en el aprendizaje Q-Learning debido a la exploración continua. La exploración permite al robot probar diferentes acciones y aprender cuál de ellas produce las mejores recompensas, aunque esto conlleve recibir recompensas bajas en ciertos episodios.

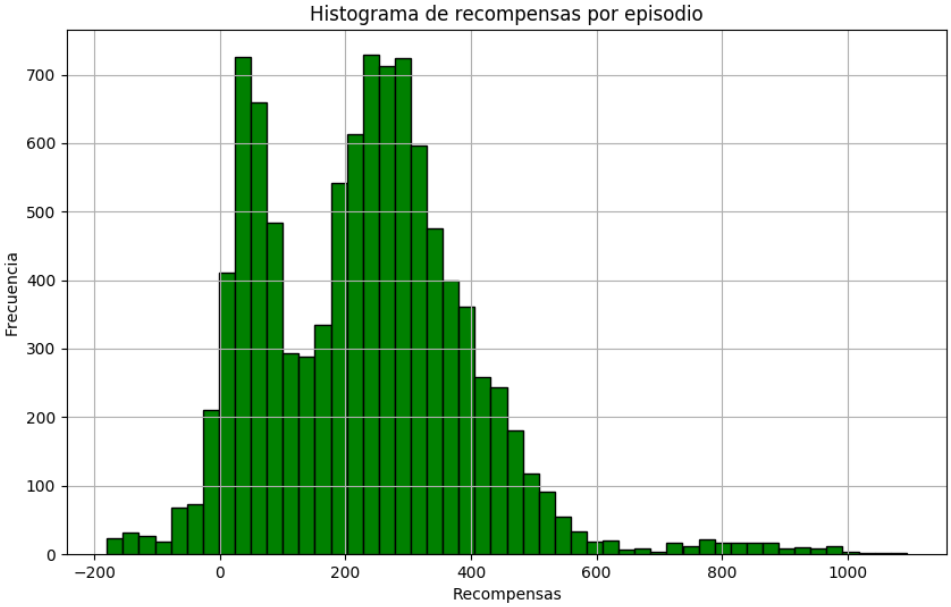


Figura 6-9. Histograma de recompensas por episodio en el primer entrenamiento con Q-Learning

El análisis del histograma de recompensas por episodio es fundamental para entender cómo se distribuyen las recompensas a lo largo de los episodios de entrenamiento. La figura 6-9, que representa gráficamente la frecuencia de las recompensas obtenidas, revela varias características importantes sobre el comportamiento del robot y su proceso de aprendizaje.

En primer lugar, la distribución de las recompensas muestra una clara bimodalidad, lo que significa que existen dos picos prominentes en el histograma. Uno de estos picos se encuentra alrededor de las recompensas bajas, cerca de 0, mientras que el otro pico se sitúa en las recompensas moderadas, alrededor de 300. La presencia de estos dos picos distintos puede indicar la existencia de dos tipos de episodios durante el entrenamiento. Por un lado, los episodios en los que el robot no fue capaz de aprender efectivamente, resultando en recompensas bajas, y, por otro lado, aquellos episodios donde el robot logró un desempeño moderado, obteniendo recompensas más altas. Esta bimodalidad refleja la variabilidad en el aprendizaje del robot, sugiriendo que mientras en algunos episodios el robot sigue explorando y ajustando su política, en otros episodios comienza a explotar estrategias más efectivas.

Además de la bimodalidad, el histograma presenta una notable asimetría, particularmente en la cola derecha de la distribución. Esta asimetría indica que, aunque raros, existen episodios en los que el robot obtiene recompensas excepcionalmente altas. La cola derecha extendida indica que hay situaciones en las que el robot encuentra estrategias muy efectivas que le permiten maximizar las recompensas significativamente más allá de lo común. Sin embargo, estos episodios son mucho menos frecuentes en comparación con los episodios donde las recompensas son bajas o moderadas. La asimetría en el histograma es un recordatorio de que, aunque el aprendizaje es mayormente incremental y gradual, puede haber momentos de aprendizaje repentino y significativo cuando el robot descubre políticas altamente eficientes.

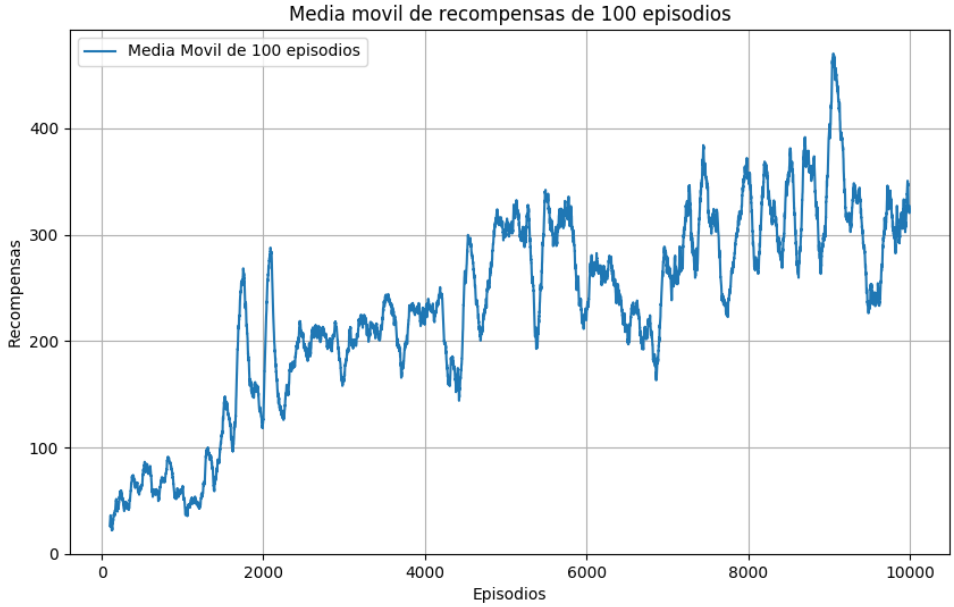


Figura 6-10. Medias móviles de las recompensas en el primer entrenamiento con Q-Learning

La figura 6-10 presenta la media móvil de las recompensas a lo largo de 100 episodios, lo cual permite suavizar la considerable variabilidad observada en el primer gráfico. Esta técnica de media móvil es particularmente útil para identificar tendencias subyacentes en series de datos con alta fluctuación, como es el caso del entrenamiento de un robot mediante Q-Learning. Al observar este gráfico, se hace evidente una serie de patrones y comportamientos del robot durante su proceso de aprendizaje.

En primer lugar, la media móvil clarifica la tendencia ascendente en el desempeño del robot. Al eliminar gran parte de la variabilidad observada en las recompensas individuales, el gráfico revela que, en general, el robot mejora su rendimiento a lo largo del tiempo. Esta tendencia ascendente es un indicador positivo de que el algoritmo de Q-Learning está funcionando de manera efectiva, permitiendo al robot aprender de sus interacciones con el entorno y optimizar sus acciones para obtener mayores recompensas. La eliminación de fluctuaciones menores y aleatorias proporciona una visión más clara del progreso del aprendizaje, lo cual es crucial para evaluar la eficacia de los parámetros de entrenamiento utilizados.

Sin embargo, a pesar de esta tendencia general positiva, se observan oscilaciones significativas en la media móvil, lo que indica que existen fluctuaciones periódicas en el desempeño del robot. Estas oscilaciones pueden deberse fundamentalmente a variaciones en la exploración y explotación de acciones. La presencia de estas fluctuaciones indica que, aunque el robot está mejorando, su rendimiento puede experimentar altibajos en función de las decisiones específicas en episodios individuales. Este comportamiento es típico en procesos de aprendizaje reforzado, donde el equilibrio entre la exploración de nuevas estrategias y la explotación de conocimientos adquiridos puede llevar a variaciones temporales en el rendimiento.

6.2.1.2. Entrenamiento 2

En este segundo entrenamiento se van a modificar los valores de los parámetros alfa y gamma, llevándolos a los extremos opuestos, los que teóricamente llevan al peor rendimiento en el aprendizaje por refuerzo. Por ello, en la segunda simulación realizada para entrenar a un robot individual utilizando el algoritmo Q-Learning, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,9	0,1	0,9	0,999

Tabla 6-2. Parámetros del algoritmo Q-Learning en el segundo entrenamiento

Un alfa alto de 0,9 significa que el nuevo valor Q tiene un impacto del 90% en la actualización, mientras que el valor existente solo tiene un impacto del 10%. Esto puede llevar a un aprendizaje muy rápido, pero también puede causar inestabilidad, ya que los valores Q pueden fluctuar considerablemente con cada nueva experiencia. Esto puede resultar en un comportamiento errático del agente, ya que se ajusta rápidamente a nuevas experiencias, sin dar suficiente peso a las experiencias pasadas.

Un gamma bajo de 0,1 indica que las recompensas futuras se valoran muy poco, con un descuento del 90% en cada paso. Esto significa que el agente se enfocará principalmente en recompensas inmediatas y tendrá una visión a corto plazo. La desventaja de un gamma tan bajo es que puede llevar al agente a tomar decisiones desfavorables, optimizando solo para beneficios a corto plazo e ignorando estrategias que podrían ser más beneficiosas a largo plazo.

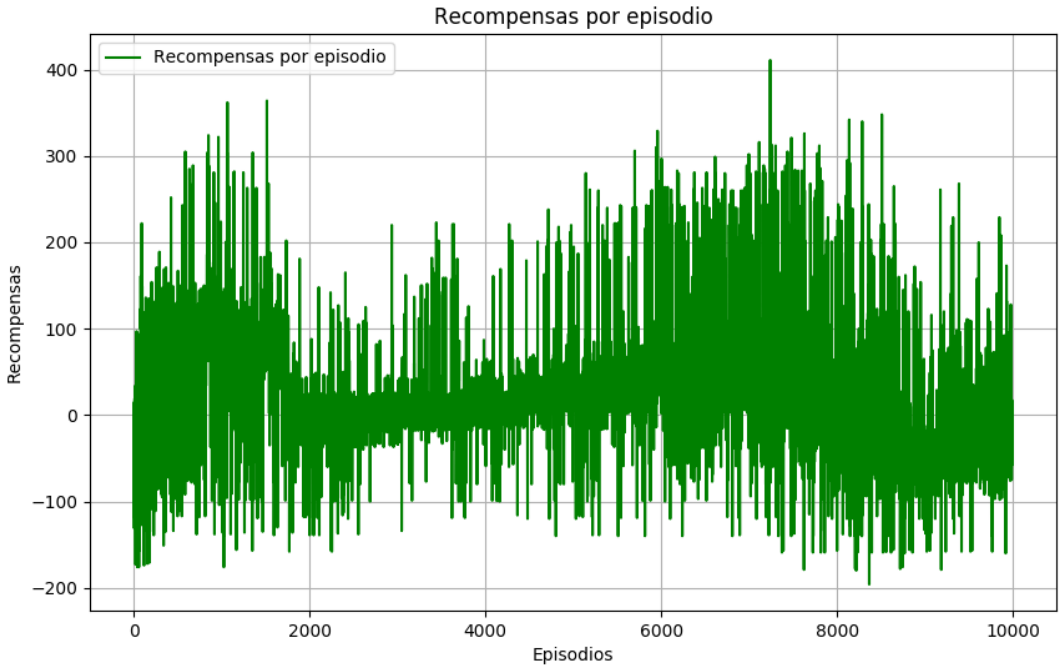


Figura 6-11. Recompensas por episodio en el segundo entrenamiento con Q-Learning

En la figura 6-11, correspondiente al segundo conjunto de parámetros, se observa que las recompensas fluctúan considerablemente y, en términos generales, se mantienen en un rango más bajo en comparación con el primer entrenamiento. Esta diferencia se hace evidente al analizar la variabilidad y la tendencia general de las recompensas, así como el enfoque del robot en la exploración y explotación de las acciones.

En cuanto a la variabilidad y la tendencia general, se destaca que la variabilidad en las recompensas es alta en ambos entrenamientos. No obstante, con el segundo conjunto de parámetros, las recompensas tienden a ser más bajas y no se observa una tendencia clara hacia la mejora. Esto contrasta notablemente con el primer entrenamiento, donde se evidenciaba una tendencia ascendente clara en las recompensas a lo largo del tiempo. La falta de una tendencia positiva en el segundo entrenamiento sugiere que el robot no está aprendiendo de manera eficiente, lo cual puede estar relacionado con los parámetros específicos utilizados.

Por otro lado, al considerar la exploración y explotación en el segundo entrenamiento, se observa que el valor

alto de alfa hace que el robot actualice agresivamente sus estimaciones de Q basándose en nuevas experiencias. Sin embargo, el valor bajo de gamma indica que el robot se enfoca más en las recompensas inmediatas en lugar de las futuras. Esta combinación de un aprendizaje agresivo con una poca valoración de las recompensas futuras parece no favorecer un aprendizaje efectivo a largo plazo. Como resultado, el robot muestra una falta de mejora sustancial en su desempeño, ya que no está acumulando experiencia de manera óptima ni desarrollando estrategias que consideren el impacto a largo plazo de sus acciones.

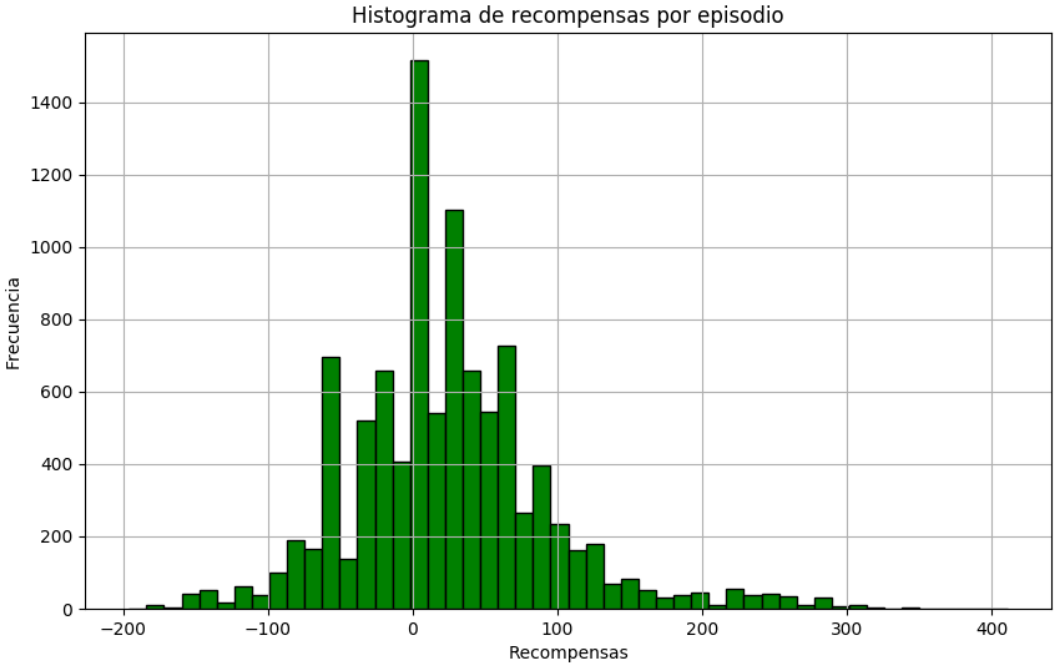


Figura 6-12. Histograma de recompensas por episodio en el segundo entrenamiento con Q-Learning

En cuanto a la distribución de recompensas, mostrada en la figura 6-12, se observa que en el segundo conjunto de parámetros la mayoría de las recompensas se agrupan alrededor de valores cercanos a 0. Esto sugiere que el robot obtiene frecuentemente recompensas bajas o nulas, lo cual indica un desempeño subóptimo. Esta tendencia puede ser consecuencia de la alta tasa de aprendizaje y el bajo factor de descuento, que hacen que el robot se enfoque demasiado en recompensas inmediatas y actualice agresivamente sus estimaciones de valor.

En contraste, el primer entrenamiento, que utilizó una configuración de alfa 0,1 y gamma 0,8, mostraba una distribución bimodal. En ese caso, además de un pico en recompensas bajas, se observaba un segundo pico alrededor de los 300 puntos de recompensa. Esta bimodalidad indicaba que, en ciertos episodios, el robot lograba un desempeño significativamente mejor, alcanzando recompensas moderadas con mayor frecuencia, lo que reflejaba un aprendizaje más efectivo y equilibrado.

Además, al examinar la frecuencia de recompensas negativas, se encuentra que hay una mayor incidencia de estas en el segundo conjunto de parámetros. Esto indica que el robot realiza más acciones subóptimas o se encuentra en situaciones desfavorables con mayor frecuencia en comparación con el primer entrenamiento. La mayor frecuencia de recompensas negativas podría estar asociada a la alta tasa de exploración inicial combinada con la alta tasa de aprendizaje. Estas configuraciones pueden llevar al robot a tomar decisiones arriesgadas o incorrectas más a menudo, resultando en un menor desempeño general.

En el primer entrenamiento, aunque también se utiliza una alta tasa de exploración inicial, el bajo valor de alfa permitía un ajuste más gradual y estable de las estimaciones de valor, lo que posiblemente contribuyera a una menor frecuencia de recompensas negativas y a un mejor desempeño general.

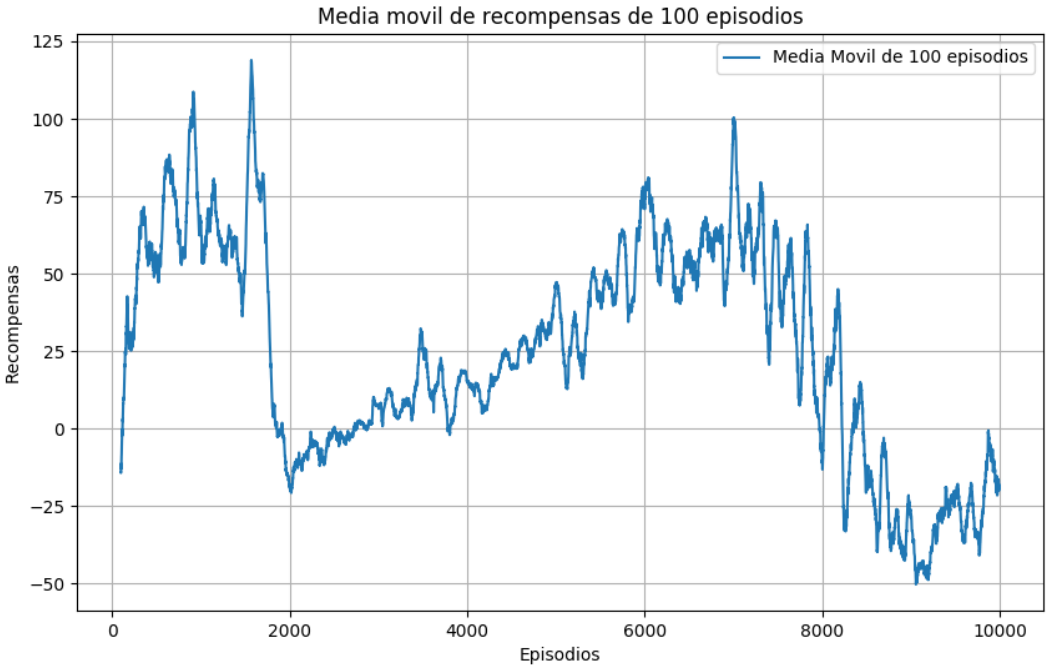


Figura 6-13. Medias móviles de las recompensas en el segundo entrenamiento con Q-Learning

En la figura 6-13 se puede comprobar que el segundo entrenamiento muestra una media móvil de recompensas en un nivel considerablemente bajo (cuatro veces menor que en el primer entrenamiento) y muy variable. Estas fluctuaciones significativas indican que el robot no está mejorando su desempeño de manera consistente a lo largo del tiempo. En comparación, la media móvil del primer entrenamiento revelaba una tendencia ascendente clara, aunque con oscilaciones periódicas. Estas oscilaciones reflejaban la variabilidad inherente en el proceso de aprendizaje, pero la tendencia general indicaba una mejora gradual y continua en el desempeño del robot. Esta comparación sugiere que los parámetros utilizados en el primer entrenamiento son más efectivos para promover el aprendizaje y la optimización del comportamiento del robot.

Respecto al impacto de los parámetros, la combinación de un alfa alto y un gamma bajo en el segundo entrenamiento parece ser menos favorable para el aprendizaje a largo plazo del robot. Un alfa alto implica que el robot actualiza agresivamente sus estimaciones de Q con base en nuevas experiencias, lo que puede resultar en inestabilidad y una falta de acumulación efectiva de conocimiento. Por otro lado, un gamma bajo significa que el robot otorga más importancia a las recompensas inmediatas en lugar de considerar las futuras, lo que puede llevar a un comportamiento miope y subóptimo.

En contraste, el primer entrenamiento con un alfa más bajo y un gamma más alto favorecía un aprendizaje más estable y una consideración más equilibrada de las recompensas a corto y largo plazo. Un alfa bajo permite una actualización más gradual y controlada de los valores de Q, mientras que un gamma alto incentiva al robot para tener en cuenta las recompensas futuras, promoviendo decisiones más estratégicas y beneficiosas a largo plazo.

6.2.1.3. Entrenamiento 3

En la tercera simulación se toman valores intermedios de alfa y gamma, con el objetivo de obtener un rendimiento intermedio entre los dos casos anteriores. Por ello, en el tercer entrenamiento realizado para entrenar a un robot individual utilizando el algoritmo Q-Learning, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,5	0,5	0,9	0,999

Tabla 6-3. Parámetros del algoritmo Q-Learning en el tercer entrenamiento

Un alfa de 0,1 en el primer entrenamiento proporcionaba un aprendizaje lento pero estable, adecuado para entornos que requieren consistencia. En el tercer entrenamiento, un alfa de 0,5 acelera el aprendizaje, lo que puede resultar en un comportamiento más rápido pero menos estable, adecuado para situaciones que demandan adaptación rápida.

Un gamma de 0,8 en el primer entrenamiento promueve un equilibrio entre recompensas inmediatas y futuras, ideal para estrategias a largo plazo. En el tercer entrenamiento, un gamma de 0,5 enfoca un poco más al robot en recompensas inmediatas, adecuado para entornos donde los beneficios a corto plazo son más relevantes, pero dando cierta importancia igualmente a las recompensas a largo plazo.

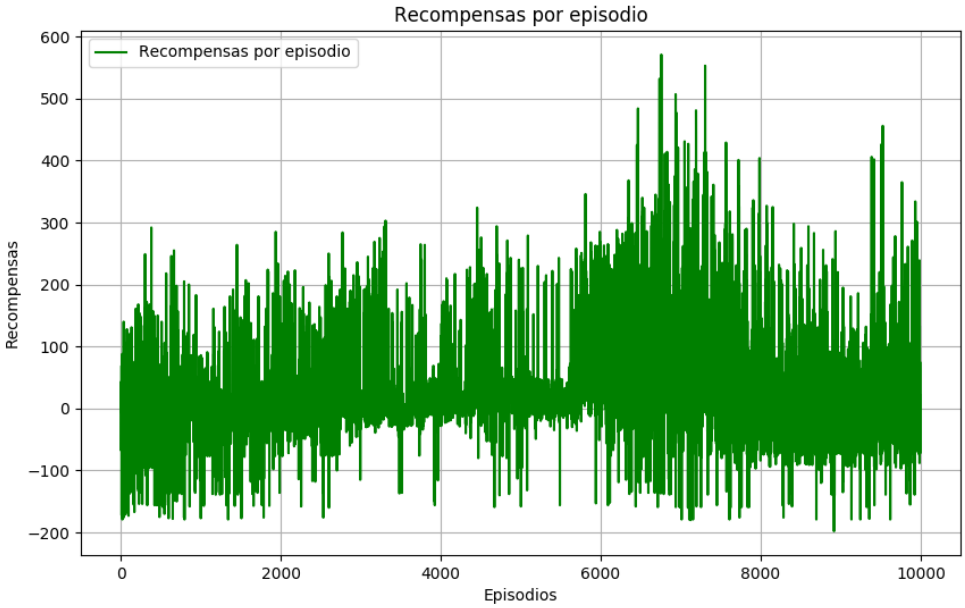


Figura 6-14. Recompensas por episodio en el tercer entrenamiento con Q-Learning

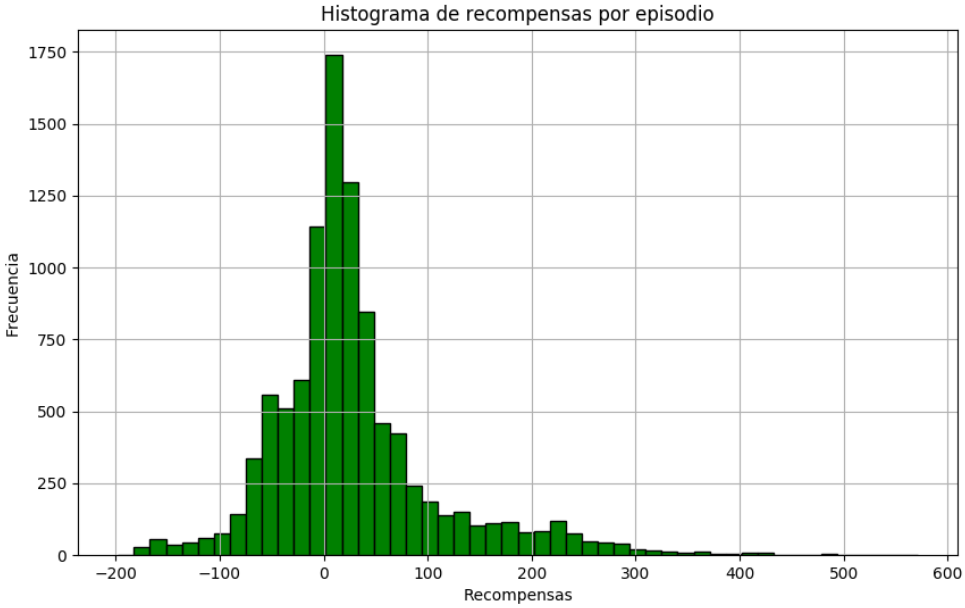


Figura 6-15. Histograma de recompensas por episodio en el tercer entrenamiento con Q-Learning

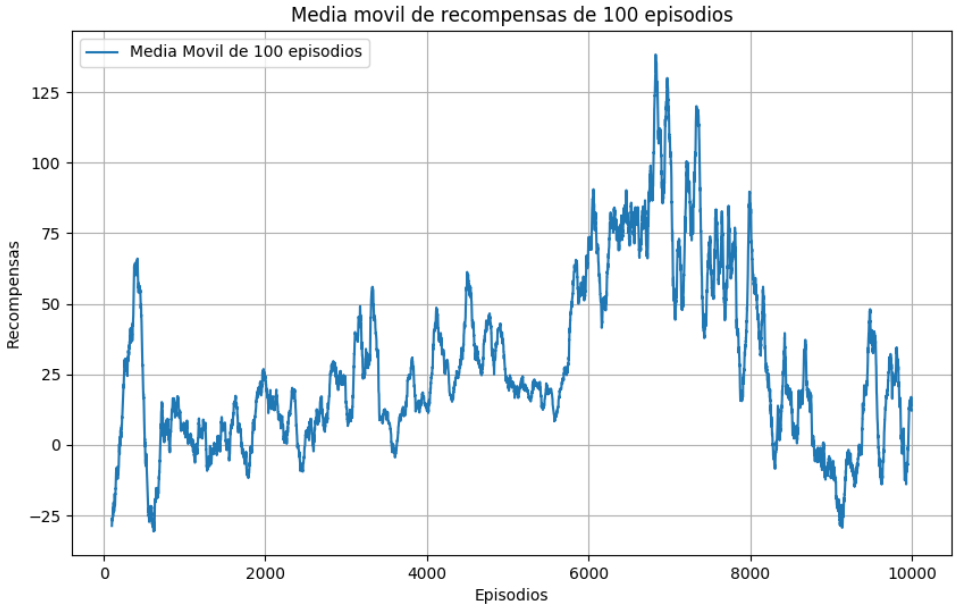


Figura 6-16. Medias móviles de las recompensas en el tercer entrenamiento con Q-Learning

Los resultados obtenidos muestran que los valores seleccionados para los parámetros alfa y gamma no son los óptimos para este escenario de entrenamiento con Q-Learning, tal y como se esperaba.

El valor de alfa, que determina la tasa de aprendizaje, resulta en una moderada sensibilidad a nuevas experiencias, lo cual es beneficioso para evitar cambios abruptos en las estimaciones Q. Sin embargo, esta moderación también implica que el agente no se adapta lo suficientemente rápido a las políticas más exitosas que va descubriendo, como se evidencia en la alta variabilidad de las recompensas y la tendencia decreciente en los episodios finales.

El valor de gamma, que pondera las recompensas futuras, al ser 0,5, implica que las recompensas inmediatas y

futuras son valoradas de manera equilibrada, pero con un descuento significativo. Este balance puede ser inadecuado en situaciones donde es crucial priorizar las recompensas a largo plazo para descubrir estrategias más robustas. La tendencia ascendente seguida de fluctuaciones y declive en las recompensas sugiere que el agente no está maximizando efectivamente las recompensas a largo plazo, posiblemente debido a una subvaloración de estas.

Los valores de alfa y gamma utilizados en este entrenamiento no son los más adecuados, ya que no permiten una adaptación rápida y eficaz a las estrategias más exitosas ni una correcta valoración de las recompensas a largo plazo.

6.2.1.4. Entrenamiento 4

En este entrenamiento y los sucesivos con Q-Learning se mantienen los valores de alfa y gamma del primer entrenamiento, ya que se trata de la combinación que mejores resultados ha dado en el aprendizaje del robot. La diferencia que se introduce es la disminución del descuento de épsilon, de cara a analizar las consecuencias que este parámetro tiene en el aprendizaje. Por ello, en la cuarta simulación realizada para entrenar a un robot individual utilizando el algoritmo Q-Learning, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,1	0,8	0,9	0,9

Tabla 6-4. Parámetros del algoritmo Q-Learning en el cuarto entrenamiento

La disminución del descuento de épsilon de 0,999 a 0,9 implica una reducción en la probabilidad de que el robot elija acciones aleatorias, lo que resulta en una transición temprana hacia la explotación de acciones conocidas. Esta disminución de épsilon, aunque facilita una convergencia más rápida a una política estable, también conlleva el riesgo de una exploración insuficiente del espacio de soluciones posibles, lo que puede impedir al robot descubrir estrategias óptimas o recompensas mayores a largo plazo. Esto compromete el rendimiento general del robot debido a la falta de exploración adecuada en las primeras etapas del entrenamiento.

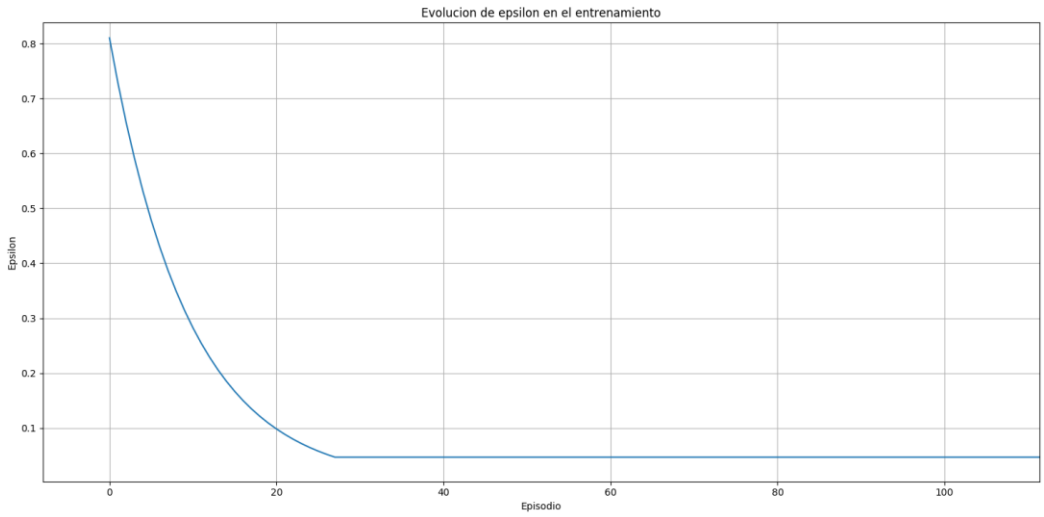


Figura 6-17. Evolución de épsilon a lo largo del entrenamiento con un decrecimiento de épsilon de 0,9

En la figura 6-17 se observa que épsilon alcanza un valor de 0,05 en cuestión de 30 episodios, lo que significa

que únicamente predomina la exploración en los 15 primeros, aproximadamente.

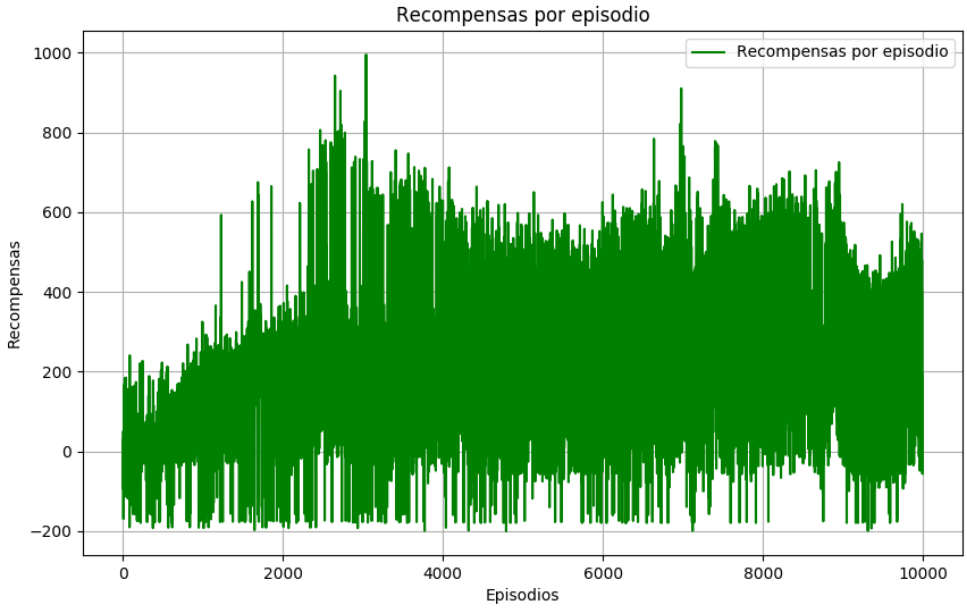


Figura 6-18. Recompensas por episodio en el cuarto entrenamiento con Q-Learning

Analizando las recompensas por episodio de la figura 6-18, se ve que estas presentan una fluctuación considerable entre valores negativos y positivos, con picos que alcanzan recompensas de 900 y una tendencia a estabilizarse en un rango entre 200 y 400, aunque con gran variabilidad y numerosos episodios con recompensas cercanas a cero o incluso negativas.

En contraste, durante el primer entrenamiento (mismos parámetros salvo el descuento de épsilon, que era de 0,999), las recompensas por episodio se mostraban más estables y alcanzaban valores más altos, con varios picos que superaban los 1000, aunque también se observaba una mayor frecuencia de episodios con recompensas cercanas a cero.

Esto sugiere que el primer entrenamiento, con una épsilon que decae más lentamente, permitía que el agente explorase más, lo que resultaba en políticas más arriesgadas que podían llevar a recompensas tanto muy altas como bajas. Por otro lado, este nuevo entrenamiento, al haber reducido la exploración debido a un mayor descuento de épsilon, ha derivado en un comportamiento más conservador del agente, con menor variabilidad en las recompensas.

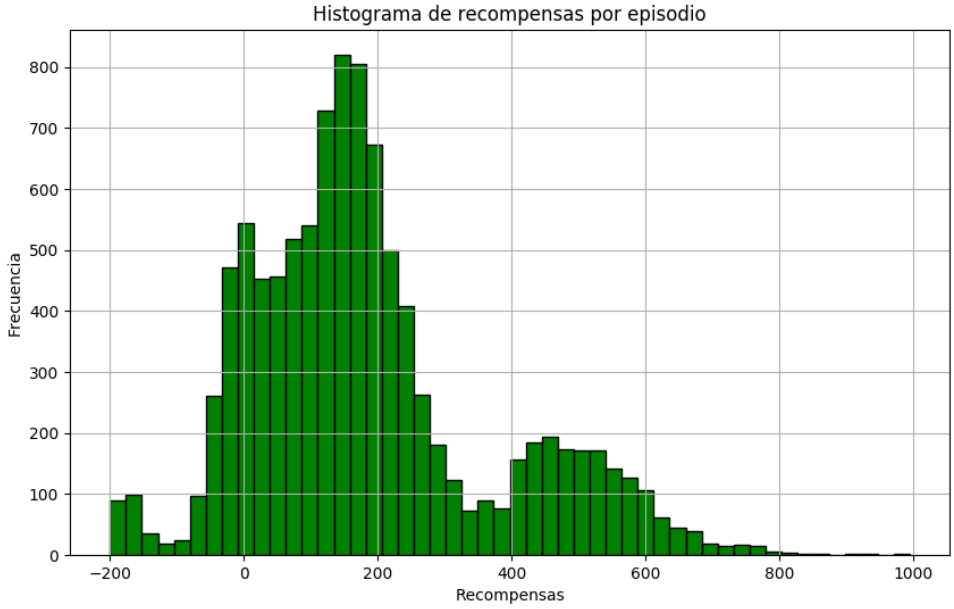


Figura 6-19. Histograma de recompensas por episodio en el cuarto entrenamiento con Q-Learning

En la figura 6-19 se observa que en este entrenamiento las recompensas tienden a concentrarse en un rango más estrecho, predominantemente entre 0 y 300, con un pico notable en torno a valores cercanos a 150. Esta distribución sugiere una menor variabilidad en las recompensas, lo que indica que el agente ha desarrollado un comportamiento más conservador.

Por otro lado, el primer entrenamiento presentaba una distribución de recompensas más amplia, con un pico principal en el rango de 0 a 200, pero también una cola considerable que se extiende hacia valores superiores a 800. Esto implicaba que el agente, en su exploración, fue capaz de alcanzar recompensas más elevadas en algunos episodios, aunque también con mayor variabilidad.

En comparación, el segundo entrenamiento ha producido un comportamiento del agente más predecible y estable, lo cual es útil en entornos donde se prefieren políticas seguras. Sin embargo, esta estabilidad podría haber limitado la capacidad del agente para descubrir y alcanzar recompensas extremadamente altas, como se observaba en el primer entrenamiento.

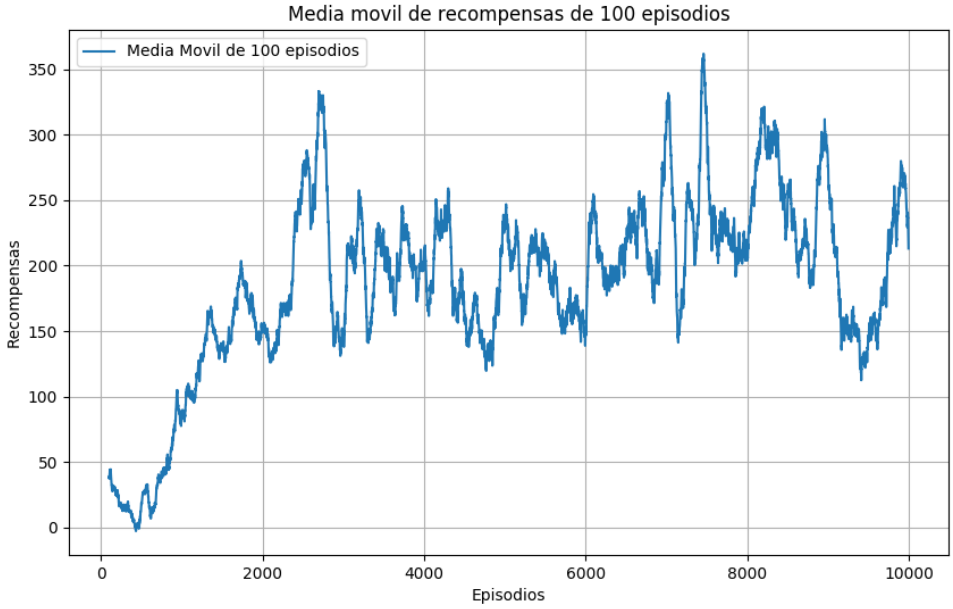


Figura 6-20. Medias móviles de las recompensas en el cuarto entrenamiento con Q-Learning

En el análisis de la media móvil de recompensas, se observa que, durante este nuevo entrenamiento, el agente ha experimentado un incremento inicial en su rendimiento hasta aproximadamente el episodio 3000. A partir de ese punto, la tendencia se ha estabilizado y ha comenzado a mostrar una ligera disminución hacia el final del entrenamiento. Este comportamiento sugiere que el agente ha encontrado una política "óptima" relativamente temprano y, al no explorar tanto en episodios posteriores, no ha logrado mejoras significativas adicionales.

En contraste, durante el primer entrenamiento, la media móvil exhibía una tendencia mucho más volátil, con mayores picos y valles, lo que reflejaba una exploración constante a lo largo del proceso. A pesar de esta volatilidad, el valor promedio de recompensas seguía una trayectoria ascendente durante más episodios y alcanzaba valores más elevados, lo que indicaba que el agente continuó ajustando y mejorando su política con el tiempo.

Al comparar ambos entrenamientos, se puede concluir que, aunque el agente en el segundo entrenamiento ha alcanzado una estabilidad más rápida, lo ha hecho a expensas de dejar de mejorar su desempeño en las etapas posteriores. Por otro lado, el primer entrenamiento, caracterizado por un descenso más gradual del parámetro épsilon, permitió que el agente mantuviera un nivel de exploración mayor, lo que resultó en una mejora continua del rendimiento, aunque con mayor variabilidad.

6.2.1.5. Entrenamiento 5

En la quinta simulación se ha reducido aún más el valor del descuento de épsilon, de manera que el valor de épsilon va a reducirse de manera bastante más significativa en cada episodio. Por ello, en el entrenamiento realizado para entrenar a un robot individual utilizando el algoritmo Q-Learning, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,1	0,8	0,9	0,8

Tabla 6-5. Parámetros del algoritmo Q-Learning en el quinto entrenamiento

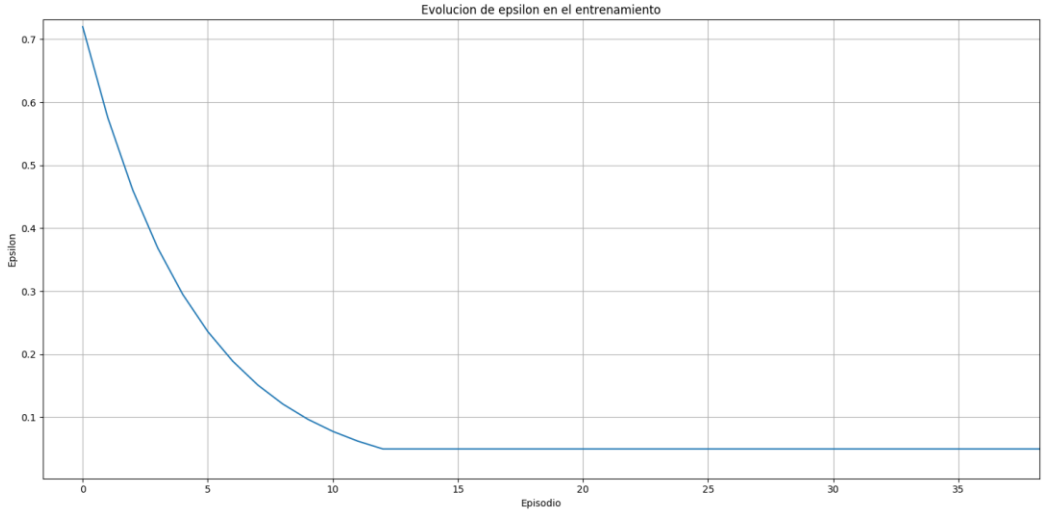


Figura 6-21. Evolución de épsilon a lo largo del entrenamiento con un decrecimiento de épsilon de 0,8

En la figura 6-21 se observa que épsilon alcanza un valor de 0,05 en cuestión de 12 episodios, lo que significa que únicamente predomina la exploración en los 6 primeros episodios, aproximadamente.

Esta reducción agresiva de épsilon puede llevar a que el robot se enfoque predominantemente en las recompensas inmediatas obtenidas al inicio del entrenamiento, lo que puede ser problemático si estas no representan la mejor estrategia a largo plazo, afectando negativamente el rendimiento final del robot. Aunque esta estrategia puede resultar en una mayor estabilidad en el comportamiento del robot durante las etapas tempranas del entrenamiento, es menos adaptable a cambios dinámicos en el entorno, ya que el robot podría aferrarse a políticas previamente aprendidas sin explorar nuevas estrategias potencialmente más efectivas.

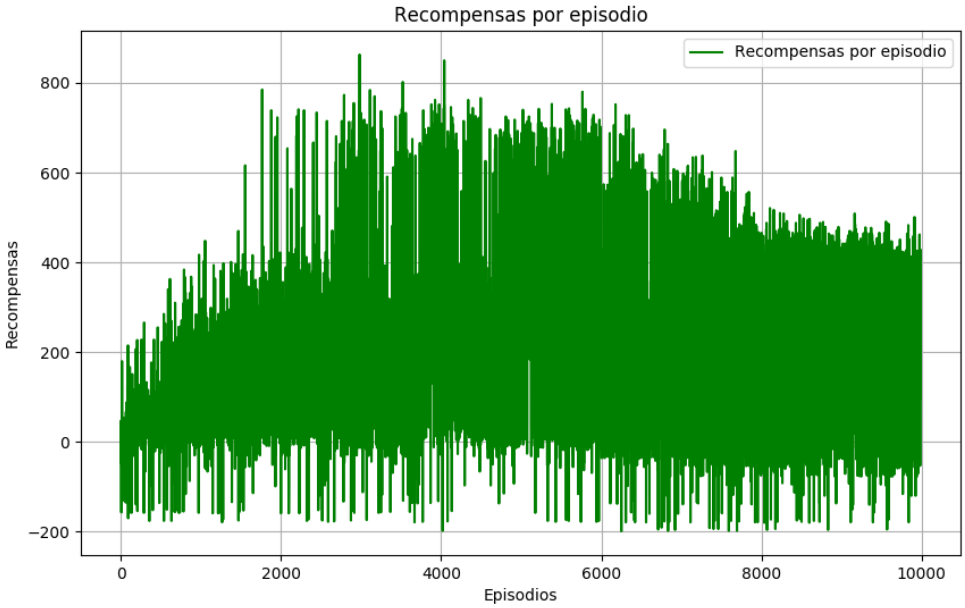


Figura 6-22. Recompensas por episodio en el quinto entrenamiento con Q-Learning

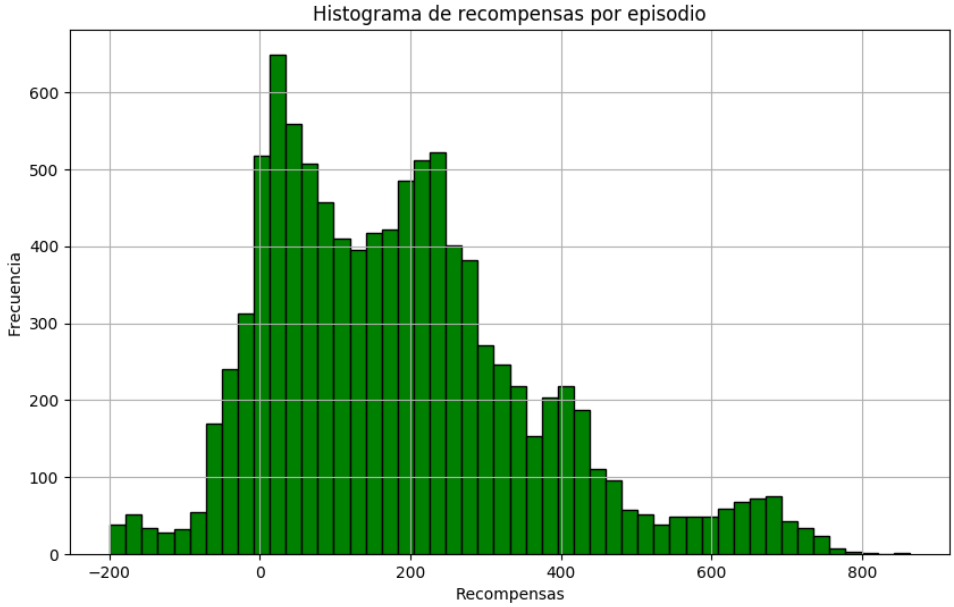


Figura 6-23. Histograma de recompensas por episodio en el quinto entrenamiento con Q-Learning

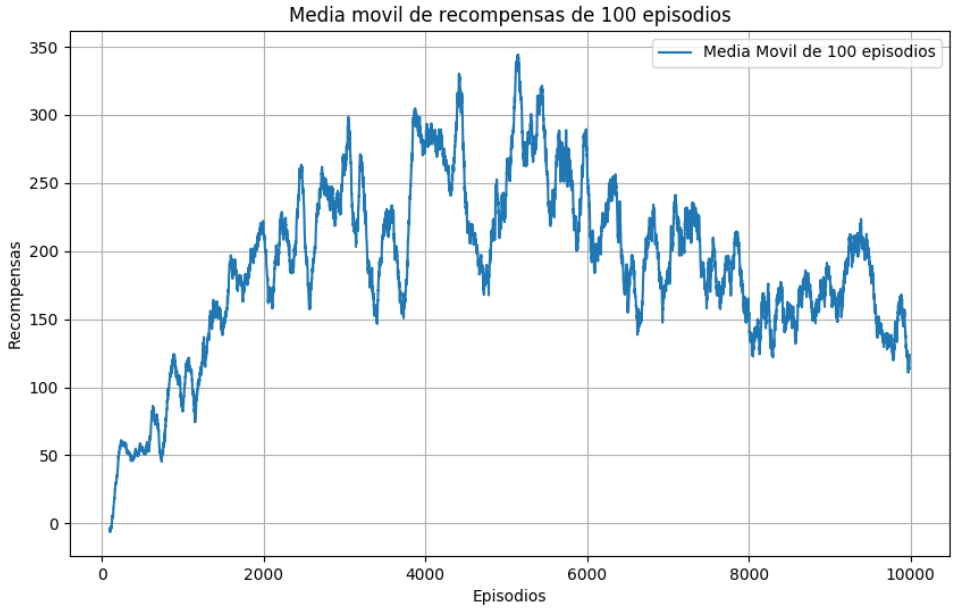


Figura 6-24. Medias móviles de las recompensas en el quinto entrenamiento con Q-Learning

Al analizar los resultados obtenidos y comparar los entrenamientos, se puede concluir que, aunque el agente en este último entrenamiento ha alcanzado una estabilidad más rápida, lo ha hecho a expensas de dejar de mejorar su desempeño en las etapas posteriores, de igual manera que ocurrió al reducir el descuento de épsilon de 0,999 a 0,9. En este caso, la estabilización se da en una fase más temprana y las recompensas obtenidas son ligeramente inferiores.

6.2.1.6. Entrenamiento 6

En la sexta simulación se ha reducido aún más el valor del descuento de épsilon. Por ello, en el entrenamiento realizado para entrenar a un robot individual utilizando el algoritmo Q-Learning, se han utilizado los siguientes

parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,1	0,8	0,9	0,7

Tabla 6-6. Parámetros del algoritmo Q-Learning en el sexto entrenamiento

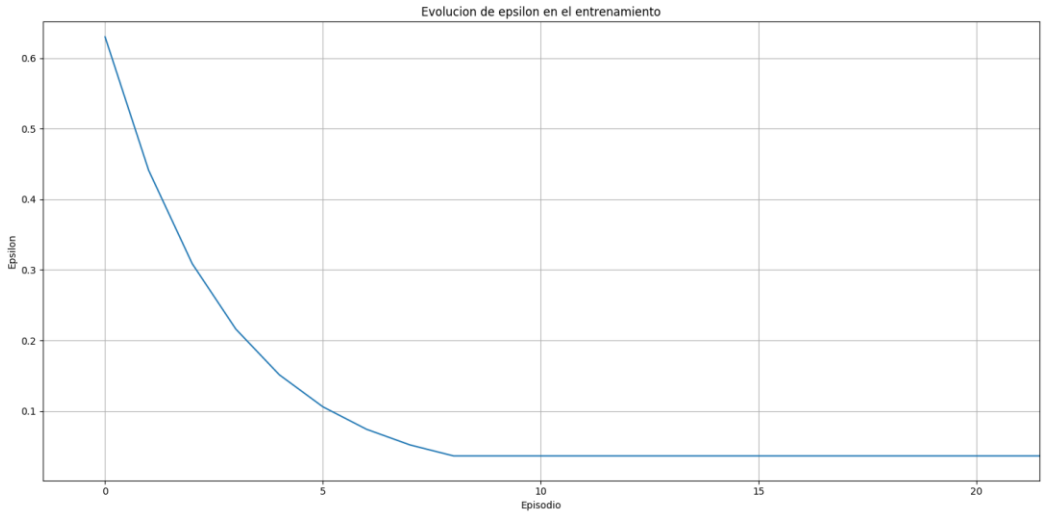


Figura 6-25. Evolución de épsilon a lo largo del entrenamiento con un decrecimiento de épsilon de 0,7

En la figura 6-25 se observa que épsilon alcanza un valor de 0,05 en cuestión de 8 episodios, lo que significa que únicamente predomina la exploración en los 4 primeros episodios, aproximadamente.

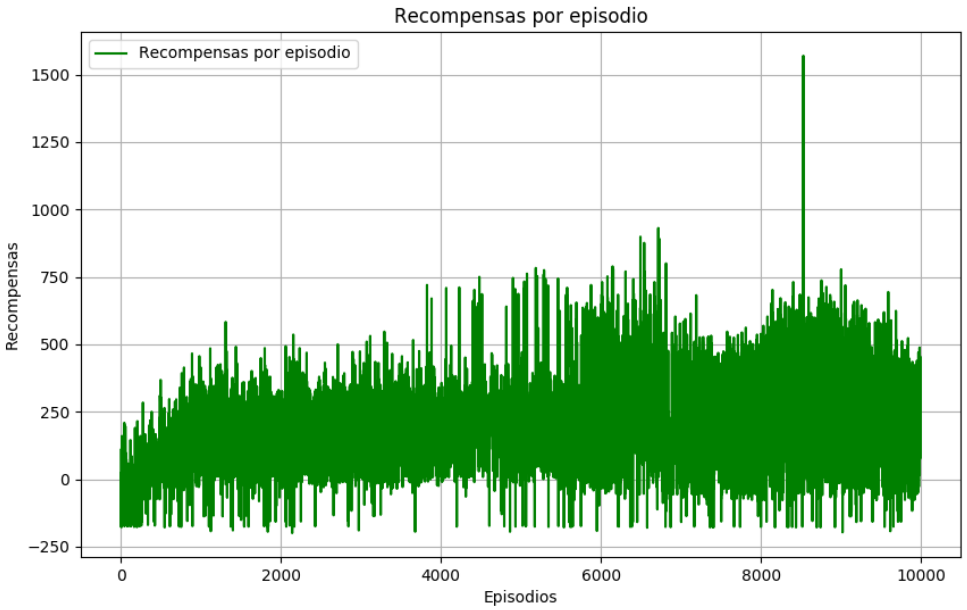


Figura 6-26. Recompensas por episodio en el sexto entrenamiento con Q-Learning

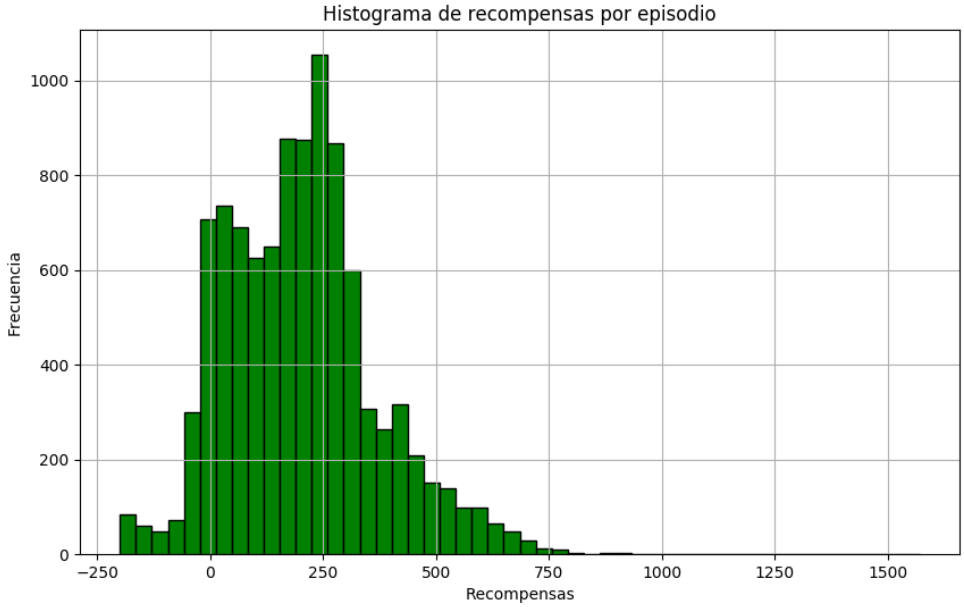


Figura 6-27. Histograma de recompensas por episodio en el sexto entrenamiento con Q-Learning

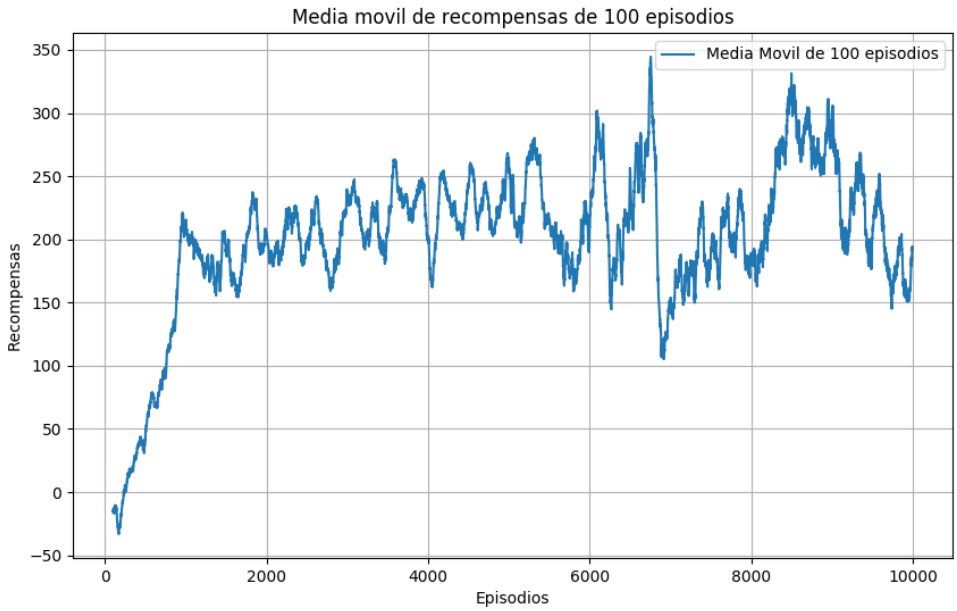


Figura 6-28. Medias móviles de las recompensas en el sexto entrenamiento con Q-Learning

Al analizar los resultados obtenidos y comparar los entrenamientos, se puede concluir que, en este caso, la estabilización se da en una fase más temprana (en torno al episodio 1000) y las recompensas obtenidas son ligeramente inferiores, estando la media móvil de estas comprendida en valores de entre 200 y 250.

Este comportamiento sigue la lógica observada al reducir el valor del descuento de épsilon en los últimos entrenamientos. Una disminución más rápida de épsilon implica una reducción de la exploración inicial y una mayor tendencia a explotar políticas que han demostrado ser exitosas. Esto puede llevar a errores potenciales debido a la falta de exploración suficiente, limitando así la capacidad de descubrir estrategias óptimas alternativas.

6.2.2. Sarsa

A continuación, se analizará el comportamiento del algoritmo Sarsa, con el objetivo de comparar su rendimiento con el observado en el algoritmo Q-Learning.

6.2.2.1. Entrenamiento 1

En la primera simulación realizada para entrenar a un robot individual utilizando el algoritmo Sarsa, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,1	0,8	0,9	0,999

Tabla 6-7. Parámetros del algoritmo Sarsa utilizados en el primer entrenamiento

Como se observó en los entrenamientos previos, estos son los valores que proporcionan un mejor rendimiento al aprendizaje por refuerzo.

Las trayectorias seguidas por el robot en cada uno de los episodios se muestran a continuación, con el objetivo de poder visualizar el comportamiento del robot antes de pasar a analizar las recompensas obtenidas.

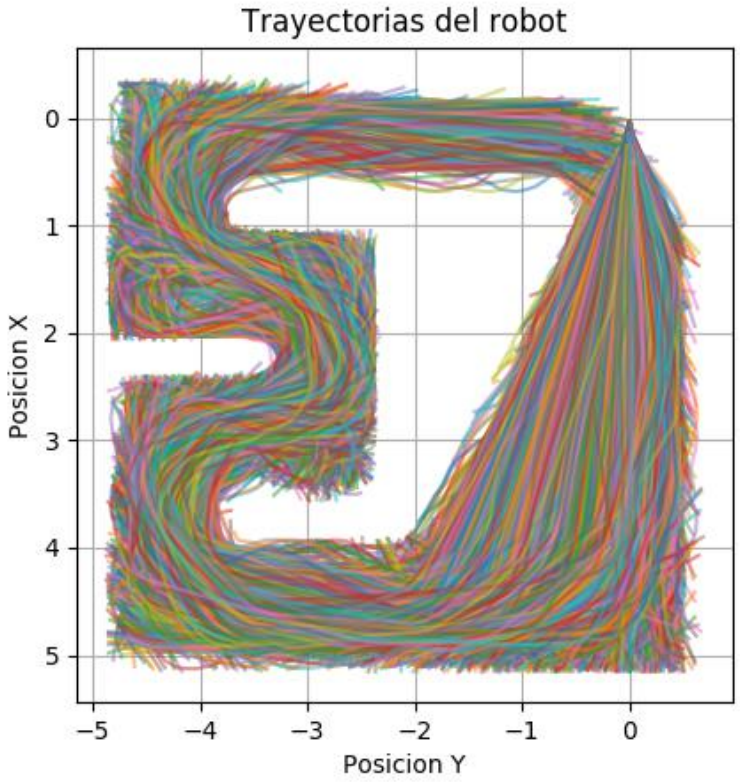


Figura 6-29. Trayectoria realizada por el robot en uno de cada diez episodios del entrenamiento con Sarsa

La figura 6-29 muestra un conjunto de trayectorias seguidas por un robot, todas comenzando desde el punto de origen en las coordenadas. Estas trayectorias son el resultado del aprendizaje del robot a través del proceso de optimización dominado por el algoritmo Sarsa.

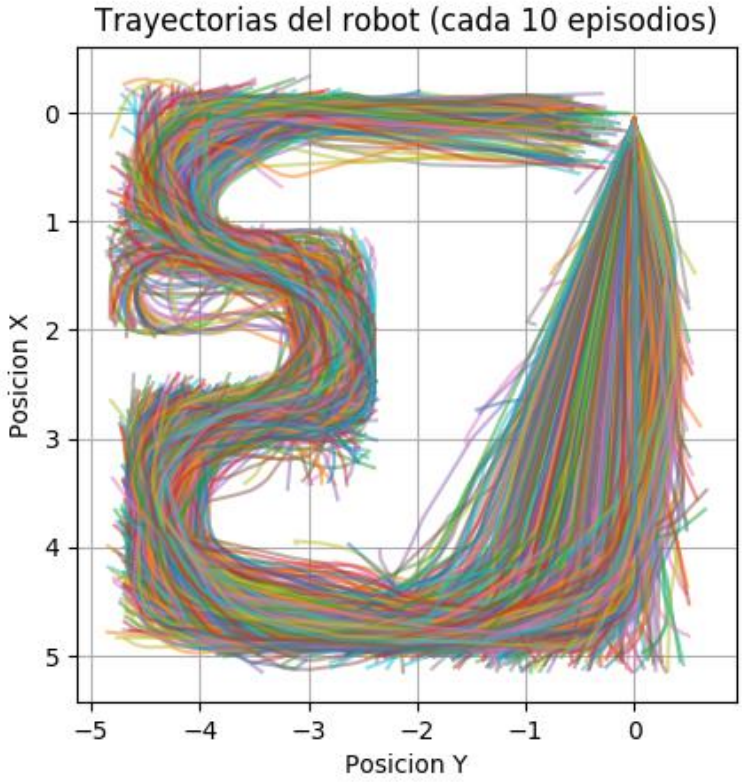


Figura 6-30. Trayectoria realizada por el robot en uno de cada diez episodios del entrenamiento con Sarsa

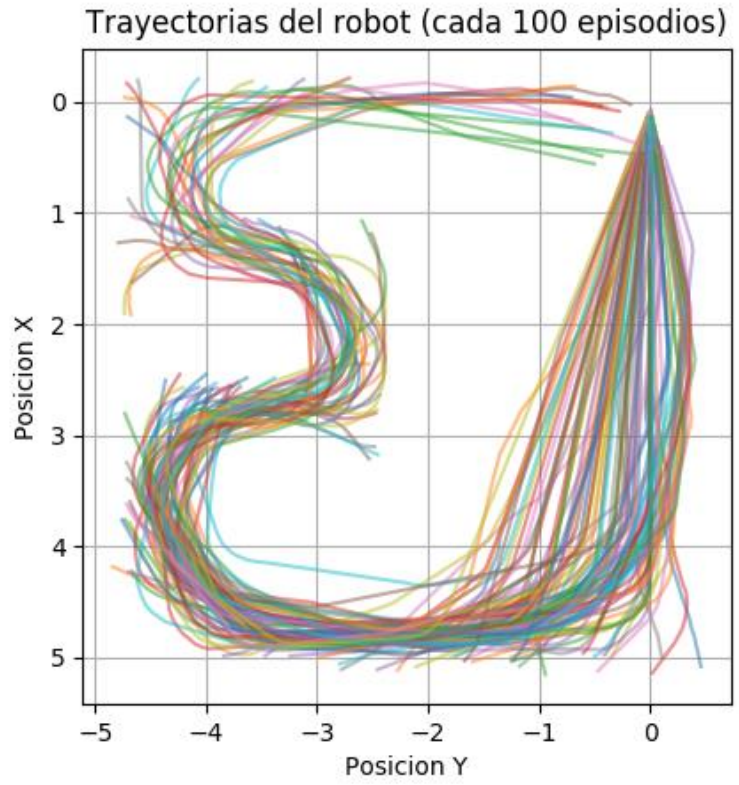


Figura 6-31. Trayectoria realizada por el robot en uno de cada cien episodios del entrenamiento con Sarsa

En estas tres imágenes, en particular en las figuras 6-30 y 6-31, que muestran las trayectorias del robot tras diez

y cien episodios, respectivamente, se puede apreciar que la cantidad de trayectorias completadas es mayor en comparación con el caso de Q-Learning. A partir de estas imágenes, es posible deducir que el rendimiento de este algoritmo será probablemente superior al del algoritmo analizado previamente.

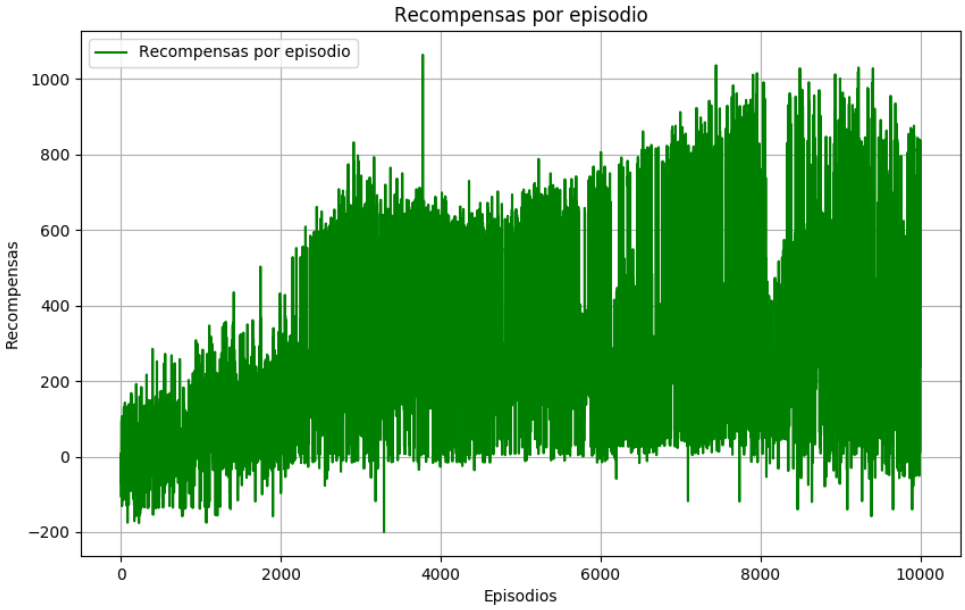


Figura 6-32. Recompensas por episodio en el primer entrenamiento con Sarsa

En la figura 6-32 se muestra un aumento gradual en las recompensas a lo largo de los 10.000 episodios. Aunque la variabilidad es alta, hay una clara tendencia ascendente, especialmente después de los primeros 2.000 episodios. Esto sugiere que el robot está mejorando progresivamente su desempeño a medida que avanza el entrenamiento. Por el contrario, el entrenamiento realizado con Q-Learning también revela una tendencia ascendente en las recompensas, pero con una mayor dispersión de estas a lo largo de todos los episodios.

En el caso del entrenamiento con Sarsa, la tendencia ascendente en las recompensas indica que el robot está aprendiendo de manera consistente y mejorando su desempeño con el tiempo. La variabilidad, aunque presente, disminuye muy ligeramente a medida que el robot aprende, lo que indica una cierta estabilización en su comportamiento. Este comportamiento ligeramente más estable puede atribuirse a la naturaleza conservadora de Sarsa, que actualiza los valores de Q basándose en las acciones realmente tomadas, lo cual favorece una convergencia más gradual y menos volátil.

Por otro lado, en el entrenamiento con Q-Learning, aunque también se observa una tendencia ascendente, la mayor dispersión de recompensas sugiere que el robot experimenta más variabilidad en su desempeño. Esta mayor variabilidad podría deberse a la naturaleza más optimista de Q-Learning, que tiende a sobrevalorar las acciones potenciales al actualizar los valores de Q basándose en las mejores acciones posibles en lugar de las acciones realmente tomadas. Esta característica de Q-Learning, mientras que puede conducir a mayores recompensas potenciales, también introduce una mayor incertidumbre y variabilidad en el proceso de aprendizaje.

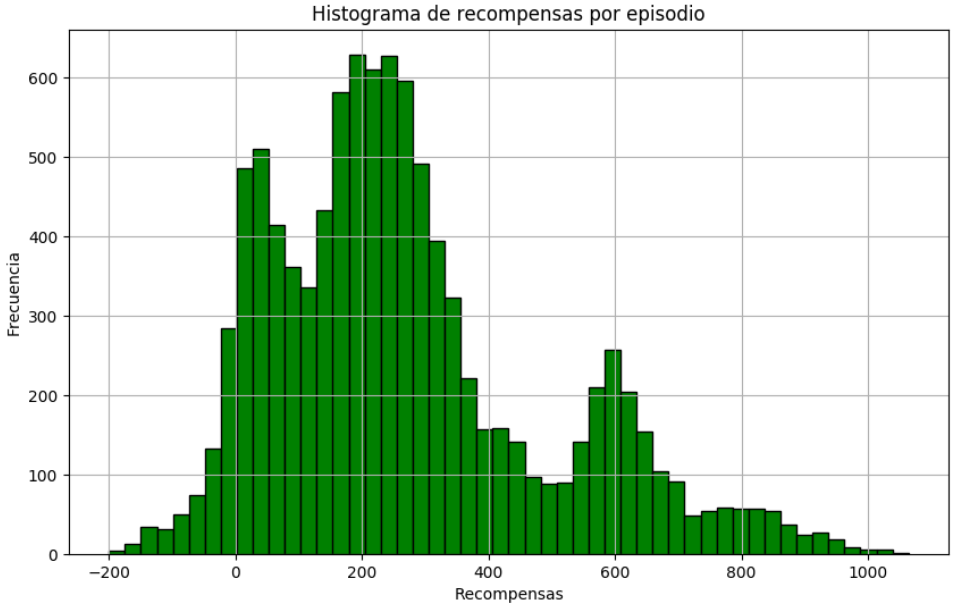


Figura 6-33. Histograma de recompensas por episodio en el primer entrenamiento con Sarsa

El histograma de recompensas obtenidas en cada episodio revela una distribución bimodal para ambos algoritmos, Sarsa y Q-Learning, pero con diferencias significativas tanto en la forma como en la dispersión de los datos. Estas diferencias proporcionan una visión clara de cómo cada algoritmo afecta el comportamiento del robot durante el proceso de aprendizaje.

Para el caso de Sarsa, en la figura 6-33 se observa que la distribución de las recompensas está más concentrada en un rango de valores que va desde 0 hasta 700, con dos picos claros alrededor de las recompensas de 200 y 600. Esta concentración indica que el robot, al utilizar el algoritmo Sarsa, tiende a obtener recompensas más consistentes y menos extremas. En otras palabras, el robot presenta un comportamiento más seguro y predecible, enfocándose en estrategias que le aseguran recompensas moderadamente altas y estables. Esto sugiere que el enfoque de Sarsa, que actualiza los valores de las acciones basándose en la política seguida, conduce a un aprendizaje más cauteloso, donde el robot evita riesgos y se adapta a comportamientos que garantizan una cierta seguridad en las recompensas obtenidas.

En contraste, los picos en la distribución de recompensas con el algoritmo Q-Learning (en 0 y 300) indican que, en la gran mayoría de situaciones, el robot experimenta con estrategias que son menos efectivas que en el caso de Sarsa, reflejando una adaptabilidad más lenta y una mayor exploración.

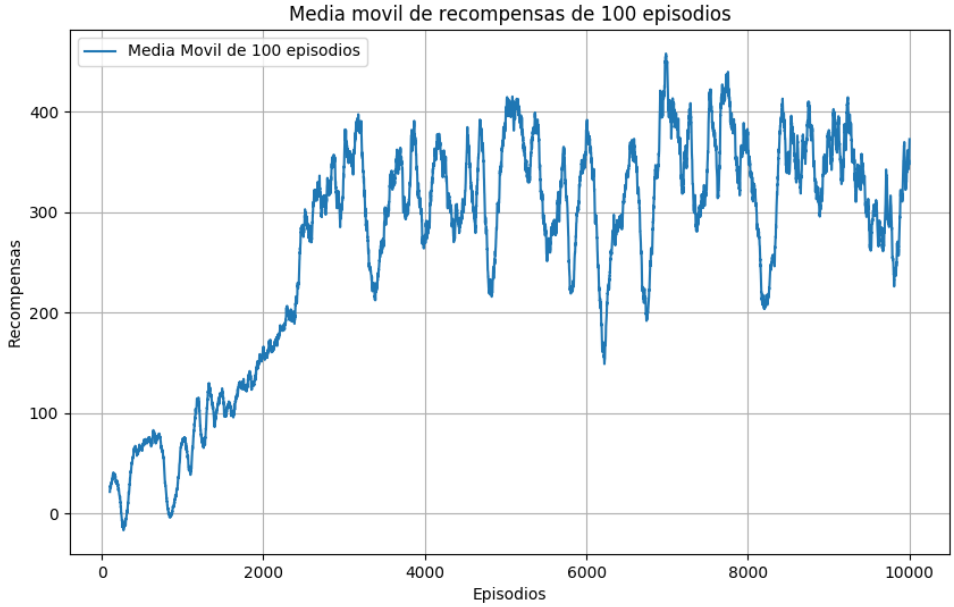


Figura 6-34. Medias móviles de las recompensas en el primer entrenamiento con Sarsa

En la figura 6-34 se puede observar que la media móvil muestra una tendencia ascendente, lo que indica una mejora continua en el desempeño del robot. Esta tendencia ascendente alcanza un pico alrededor del episodio 7.000, y se mantiene relativamente alta hasta el final del entrenamiento. Esto sugiere que el robot, una vez alcanzado un nivel óptimo de desempeño, logra mantenerlo de manera estable. Además, las oscilaciones en la media móvil de Sarsa son menos pronunciadas que en el caso de Q-Learning, lo cual indica una mayor estabilidad en el desempeño del robot. La menor variabilidad sugiere que el robot no solo mejora su rendimiento, sino que también lo hace de manera más consistente y confiable a lo largo del tiempo.

Por otro lado, en el entrenamiento con Q-Learning, la tendencia ascendente en la media móvil también es evidente, lo que muestra que el robot está mejorando su desempeño a lo largo del tiempo. Sin embargo, las oscilaciones son más pronunciadas, especialmente después del episodio 7.000. Estas fluctuaciones más marcadas sugieren que, aunque el robot mejora su desempeño, lo hace de manera menos consistente y con más altibajos comparado con Sarsa. La naturaleza exploradora de Q-Learning, que tiende a probar nuevas acciones y estrategias en busca de mejores recompensas, podría ser la causa de estas oscilaciones.

6.2.2.2. Entrenamiento 2

En la segunda simulación se han escogido la combinación de parámetros que peor rendimiento proporciona al aprendizaje por refuerzo. Por ello, en el segundo entrenamiento realizado para entrenar a un robot individual utilizando el algoritmo Sarsa, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,9	0,1	0,9	0,999

Tabla 6-8. Parámetros del algoritmo Sarsa utilizados en el segundo entrenamiento

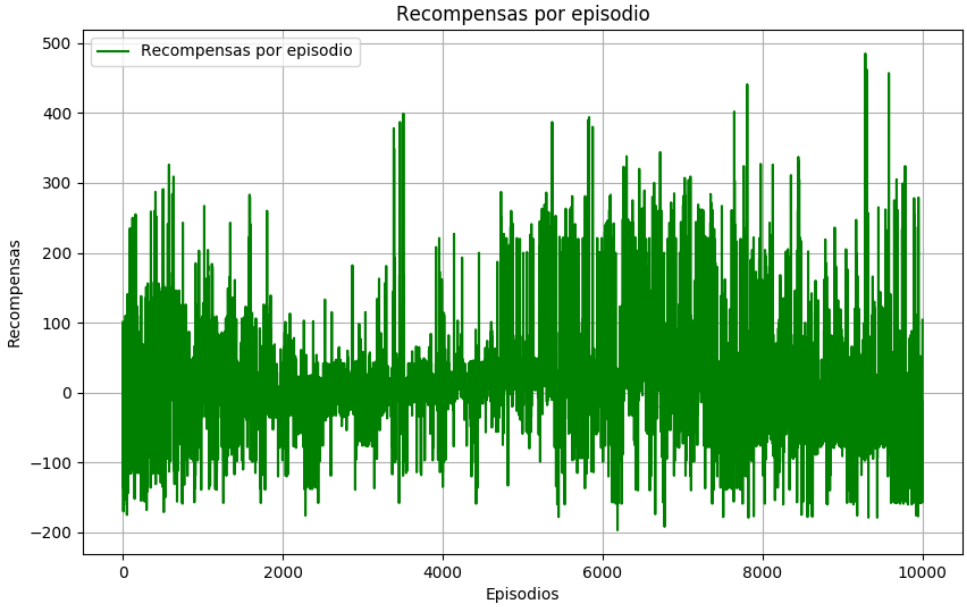


Figura 6-35. Recompensas por episodio en el segundo entrenamiento con Sarsa

En el primer entrenamiento, el gráfico de recompensas por episodio utilizando el algoritmo Sarsa mostraba un aumento gradual y consistente en las recompensas a lo largo de los 10.000 episodios. Al inicio del entrenamiento, las recompensas obtenidas por el robot eran bajas, lo cual es típico en las primeras fases de aprendizaje donde el agente está aún explorando y probando diferentes acciones sin una estrategia clara. Sin embargo, a medida que avanzaba el entrenamiento, se observaba una tendencia ascendente clara en las recompensas, lo que indicaba una mejora continua y sostenida en el desempeño del robot. Esta mejora gradual ponía en evidencia que el robot estaba aprendiendo de manera efectiva a optimizar sus acciones para maximizar las recompensas a lo largo del tiempo.

En contraste, en el segundo entrenamiento, donde se ha empleado un alfa más alto y un gamma más bajo, las recompensas por episodio que aparecen en la figura 6-35 muestran una mayor variabilidad y una tendencia que no va en ascenso. Aunque en algunos episodios se alcanzan recompensas altas, estos picos no son representativos de una mejora continua y sostenida como en el primer caso. La mayor variabilidad en las recompensas indica que el robot experimenta más altibajos en su desempeño, con episodios que alternan entre recompensas altas y bajas. Este comportamiento podría atribuirse a las actualizaciones más agresivas de los valores de Q debido a un alfa alto, lo que puede llevar a decisiones menos estables y a una sobrevaloración de las recompensas inmediatas debido a un gamma bajo.

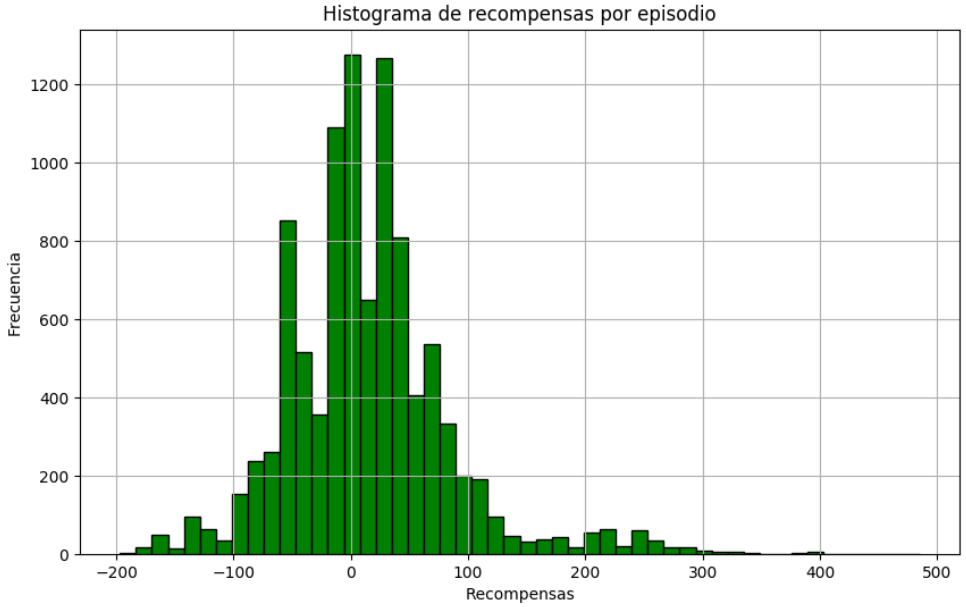


Figura 6-36. Histograma de recompensas por episodio en el segundo entrenamiento con Sarsa

El histograma de recompensas del primer entrenamiento mostraba una distribución bimodal con picos en 200 y 600, indicando que el robot, con un alfa de 0,1 y un gamma de 0,8, había aprendido a obtener recompensas consistentes en esos rangos. Estos parámetros favorecían un aprendizaje gradual y la consideración de recompensas futuras, resultando en una estrategia efectiva y estable.

En contraste, el histograma del segundo entrenamiento, representado en la figura 6-36, presenta una concentración notablemente más alta de recompensas cercanas a cero. Esta diferencia en la distribución sugiere que con los actuales parámetros el robot experimenta un peor desempeño, llevando a cabo episodios sin éxito en la mayoría de los casos. Esta combinación de parámetros parece provocar que el robot no desarrolle estrategias consistentes, resultando en un mayor número de episodios con recompensas bajas o incluso negativas.

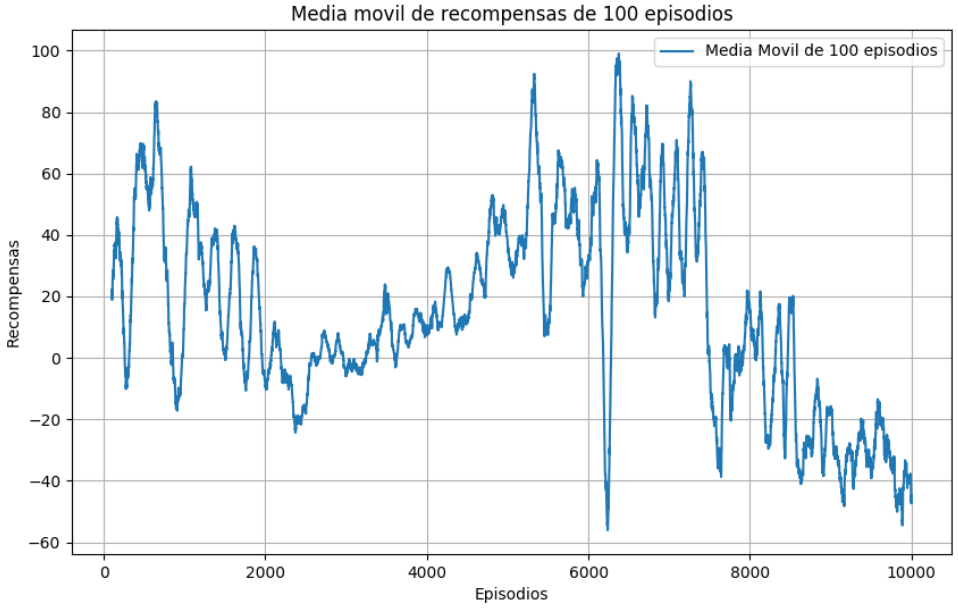


Figura 6-37. Medias móviles de las recompensas en el segundo entrenamiento con Sarsa

En el primer entrenamiento, la media móvil mostraba una tendencia ascendente consistente, alcanzando picos alrededor del episodio 7.000 y manteniéndose relativamente alta hasta el final del entrenamiento.

Por el contrario, en el segundo entrenamiento la media móvil de recompensas muestra una tendencia menos constante y que se vuelve descendente al final del entrenamiento. Inicialmente se observan picos en las recompensas promedio, lo que indica momentos de mejor desempeño. Sin embargo, estos picos son seguidos por una disminución en las recompensas promedio hacia el final del entrenamiento, lo que sugiere que el robot no logra mantener el nivel de desempeño alcanzado inicialmente. Las oscilaciones son más pronunciadas en este caso, indicando una mayor inestabilidad en el desempeño del robot. La alta variabilidad y la falta de una tendencia ascendente clara sugieren que los parámetros alfa alto y gamma bajo no son tan efectivos para fomentar un aprendizaje estable y sostenido.

Estas diferencias en la media móvil de recompensas reflejan el impacto significativo de los parámetros de aprendizaje en el desempeño del robot. En el primer entrenamiento, un alfa de 0,1 y un gamma de 0,8 permiten actualizaciones más controladas y una mayor consideración de las recompensas futuras, lo que favorece un aprendizaje gradual y estable. En cambio, en el segundo entrenamiento, un alfa alto de 0,9 implica actualizaciones más agresivas, mientras que un gamma bajo de 0,1 reduce la consideración de las recompensas futuras, llevando a un aprendizaje más errático y menos predecible.

Comparando los resultados obtenidos con los resultados extraídos del entrenamiento con Q-Learning usando los mismos valores de parámetros, es evidente que el comportamiento es prácticamente idéntico. En ambos casos, el agente no tiene la capacidad de optimizar una política robusta a largo plazo y termina en un estancamiento.

6.2.2.3. Entrenamiento 3

En la tercera simulación se tomará un valor intermedio de los parámetros alfa y gamma. Por ello, en el tercer entrenamiento realizado para entrenar a un robot individual utilizando el algoritmo Sarsa, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,5	0,5	0,9	0,999

Tabla 6-9. Parámetros del algoritmo Sarsa utilizados en el tercer entrenamiento

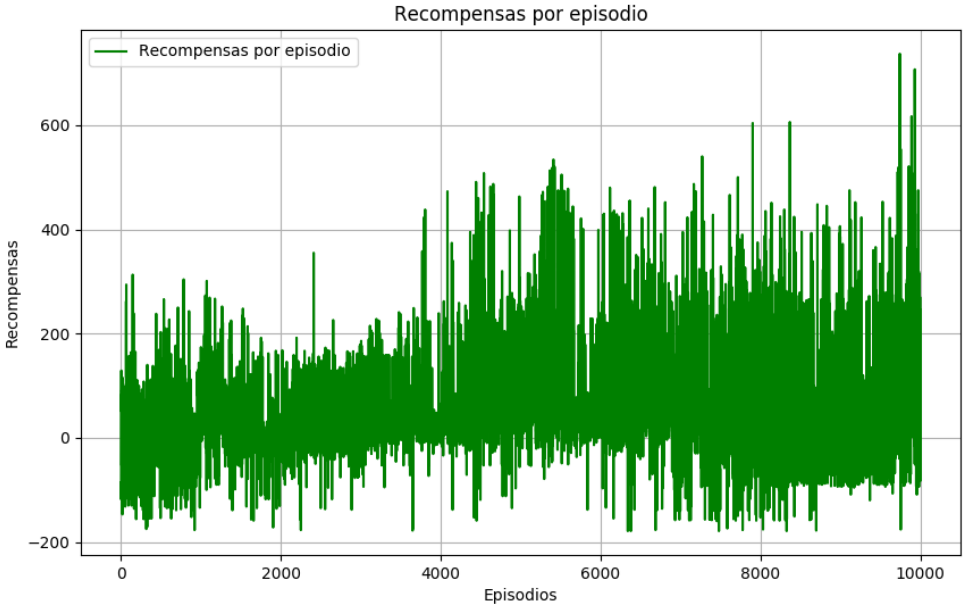


Figura 6-38. Recompensas por episodio en el tercer entrenamiento con Sarsa

Figura 6-39. Histograma de recompensas por episodio en el tercer entrenamiento con Sarsa

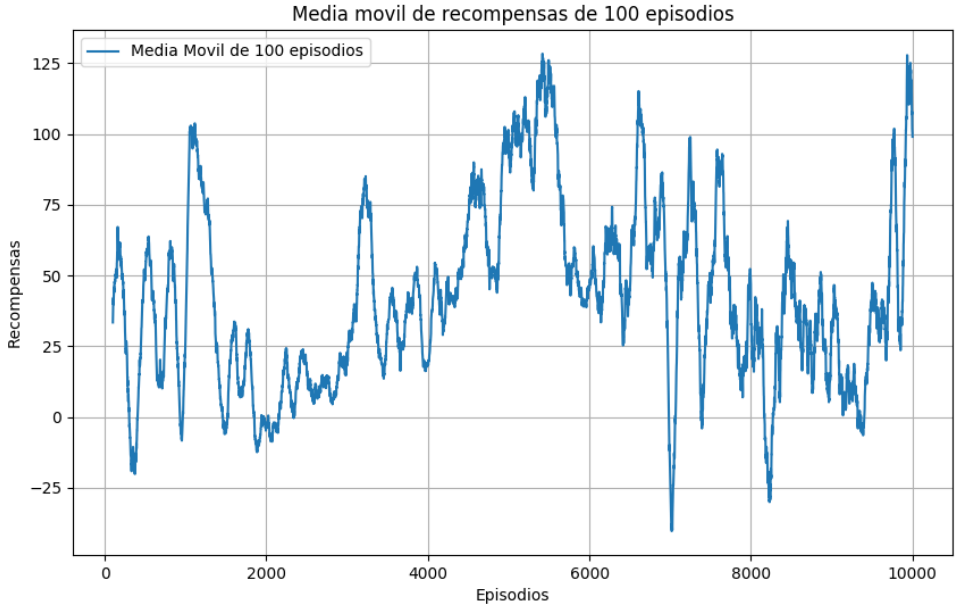


Figura 6-40. Medias móviles de las recompensas en el tercer entrenamiento con Sarsa

Los resultados obtenidos muestran que los valores seleccionados para los parámetros alfa y gamma no son los ideales para este escenario de entrenamiento con el algoritmo Sarsa, tal y como se preveía.

La sensibilidad moderada a nuevas experiencias introducida por el valor de alfa implica que el agente no se ajusta con suficiente rapidez a las políticas más exitosas que va descubriendo, como se evidencia en la alta variabilidad de las recompensas y la tendencia decreciente en los episodios finales.

La tendencia ascendente seguida de fluctuaciones y declive en las recompensas sugiere que el agente no está maximizando efectivamente las recompensas a largo plazo, posiblemente debido a una subvaloración de las recompensas futuras.

Los valores de alfa y gamma utilizados en este entrenamiento no son los más apropiados, ya que no permiten una adaptación rápida y efectiva a las estrategias más exitosas ni una correcta valoración de las recompensas a largo plazo.

Comparando los resultados obtenidos con los resultados extraídos del entrenamiento con Q-Learning usando los mismos valores de parámetros, se observa que el rendimiento con Sarsa es ligeramente superior que con Q-Learning. Con Sarsa se alcanza un promedio de recompensas más elevado a lo largo de todo el entrenamiento, lo que se traduce en un aprendizaje más robusto y consistente. Esto indica que el enfoque de Sarsa, que ajusta los valores de las acciones según la política seguida, resulta en un aprendizaje más prudente. De esta manera, el robot tiende a evitar riesgos y se adapta a comportamientos que aseguran un nivel consistente de recompensas.

6.3. Entrenamiento de varios robots con obstáculos móviles

Se realizará el entrenamiento de varios robots con obstáculos móviles, donde dos robots deben trabajar juntos de manera simultánea. Al igual que con el entrenamiento individual, se aplicarán los algoritmos de Q-Learning y Sarsa para evaluar cómo la presencia de obstáculos móviles, en este caso robots, influye en la eficacia del aprendizaje.

El objetivo es identificar cómo la interacción entre múltiples robots puede mejorar o complicar el proceso de aprendizaje.

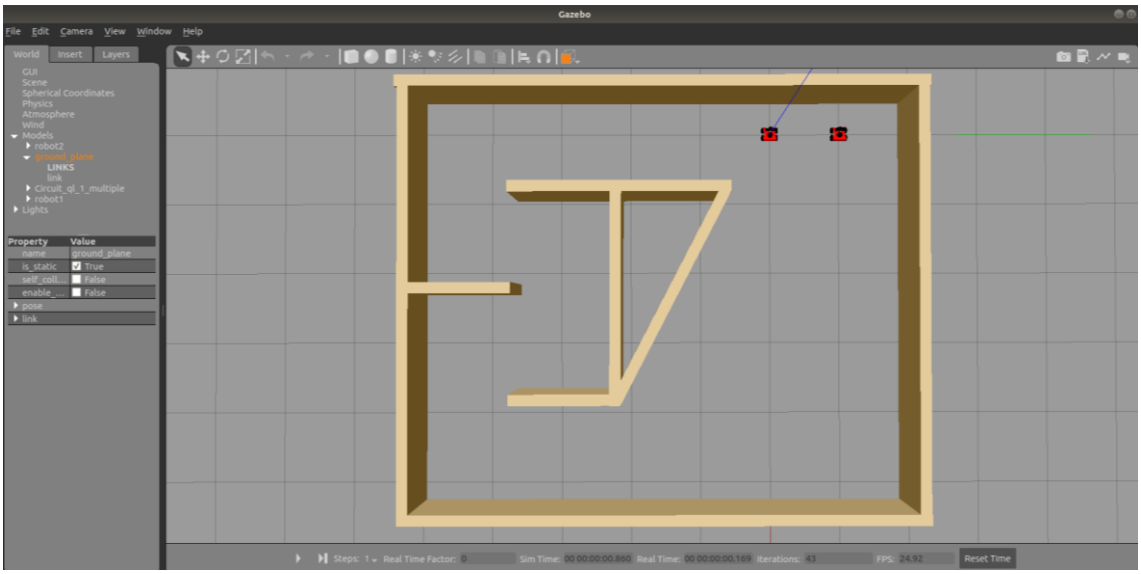


Figura 6-41. Entorno de simulación multi-robot

En estos entrenamientos se mostrarán los resultados obtenidos para uno de los dos robots que cooperan, dado que las recompensas recibidas por ambos son prácticamente idénticas en términos generales.

6.3.1. Q-Learning

6.3.1.1. Entrenamiento 1

En la primera simulación multi-robot se emplearán los parámetros que mejores resultados han dado en los entrenamientos previos. Por ello, en la primera prueba realizada para entrenar a un sistema de dos robots utilizando el algoritmo Q-Learning, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,1	0,8	0,9	0,999

Tabla 6-10. Parámetros del algoritmo Q-Learning utilizados en el primer entrenamiento multi-robot

Las trayectorias seguidas por los dos robots en cada uno de los episodios se muestran a continuación, con el objetivo de poder visualizar el comportamiento de los agentes antes de pasar a analizar las recompensas obtenidas.

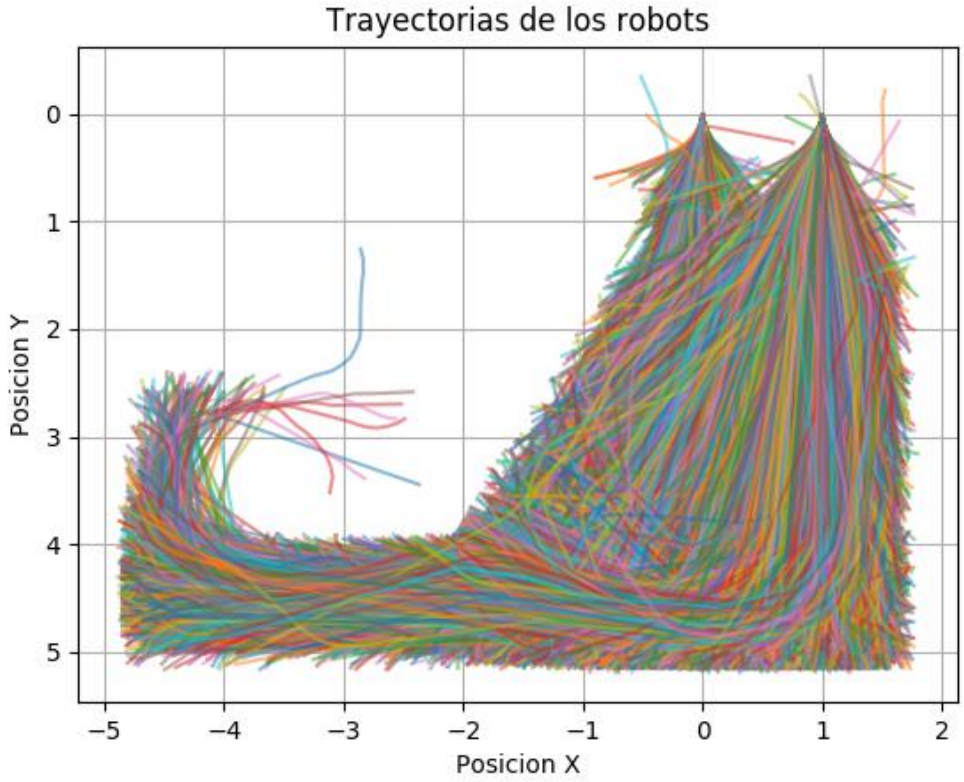


Figura 6-42. Trayectorias de los robots en cada episodio del entrenamiento múltiple con Q-Learning

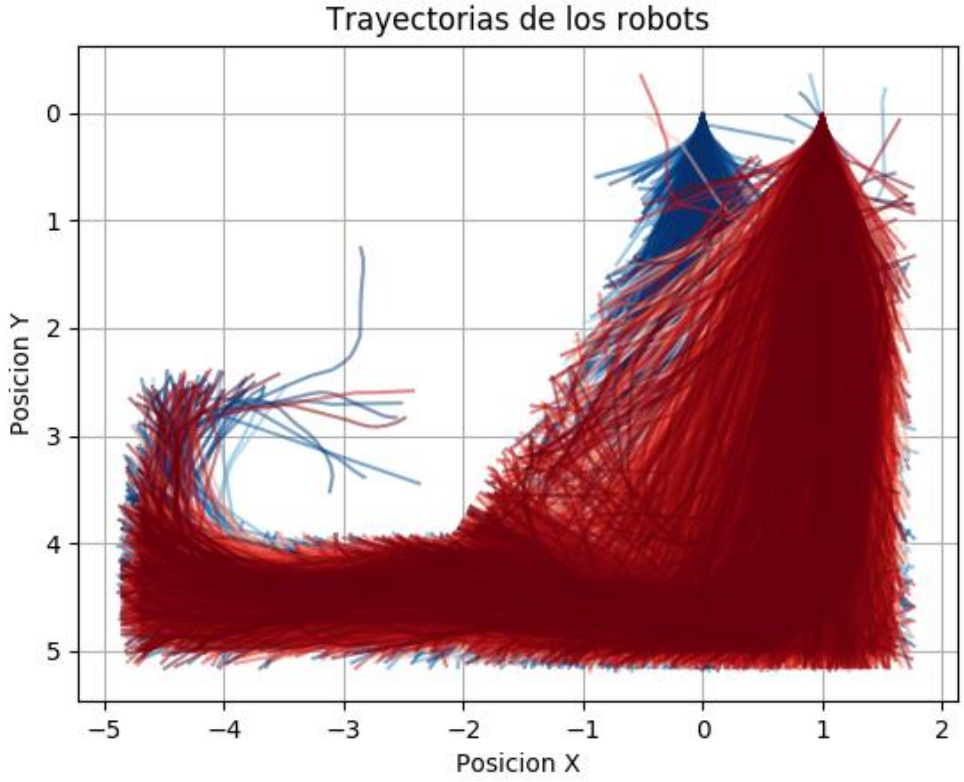


Figura 6-43. Trayectorias de los robots en cada episodio del entrenamiento múltiple con Q-Learning por colores

En la figura 6-42 se muestran las trayectorias llevadas a cabo por cada uno de los robots a lo largo del entrenamiento. A diferencia del caso de un único robot, este revela una mayor complejidad en las rutas exploradas, y es fácilmente apreciable que en ningún caso se llega a completar el laberinto. Esta dificultad para completar el laberinto podría ser resultado de la falta de espacio en el tramo donde terminan sus trayectorias, punto en el que ambos robots intentan avanzar a través de un área limitada. Esto provoca que, al intentar moverse, terminen chocando ya sea con las paredes o entre ellos, impidiéndoles progresar. Para confirmar esta hipótesis, se llevará a cabo una prueba adicional utilizando un recorrido más amplio.

La complejidad de este entrenamiento surge de la interacción entre los dos robots, que deben tener cuidado con los movimientos del otro robot para moverse en el espacio compartido, lo que introduce variabilidad adicional en las trayectorias. Las líneas en la figura indican las diferentes rutas seguidas por los robots, y su dispersión refleja no solo el proceso de aprendizaje individual, sino también la necesidad de adaptarse a las acciones del otro robot.

La figura 6-43, así como las figuras 6-45 y 6-47 que se muestran a continuación, revelan diferentes tonos (azul y rojo) para las trayectorias seguidas por cada robot. Esto se hace con el propósito de verificar que ambos agentes desarrollan políticas de movimiento prácticamente idénticas a medida que avanzan los episodios.

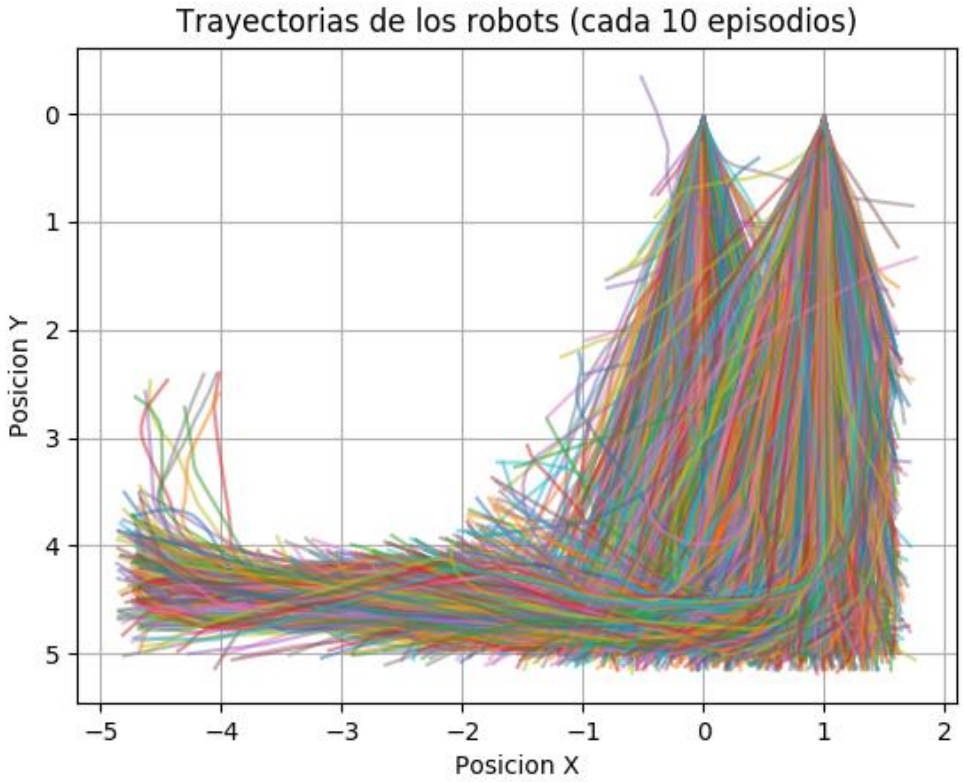


Figura 6-44. Trayectorias de los robots en uno de cada cien episodios del entrenamiento múltiple con Q-Learning

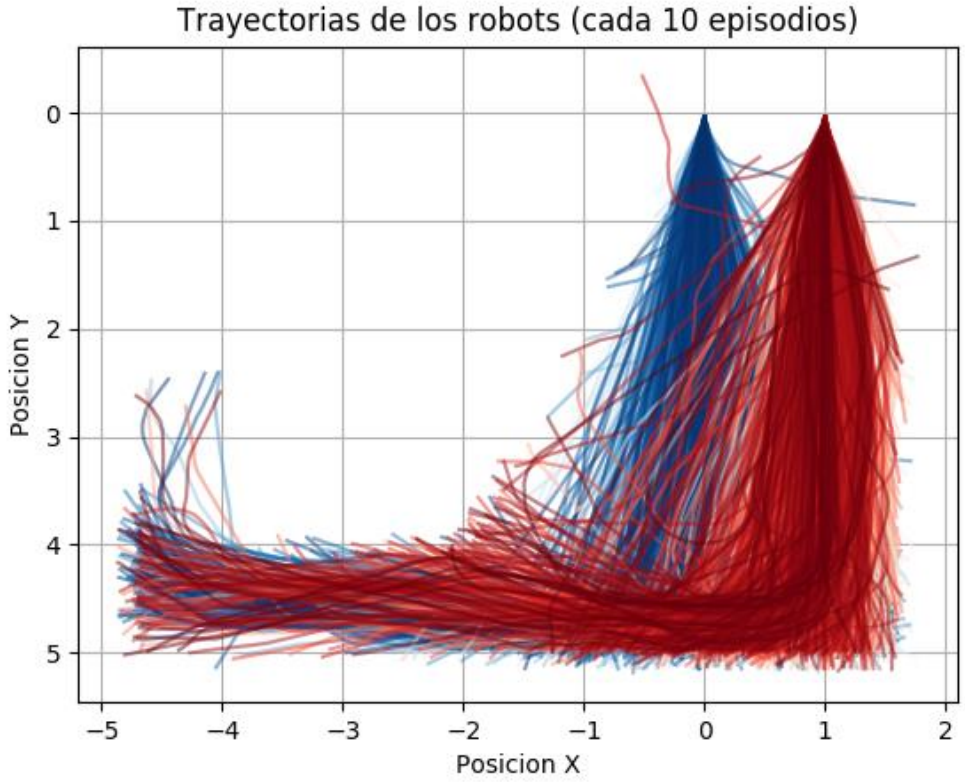


Figura 6-45. Trayectorias de los robots en uno de cada cien episodios del entrenamiento múltiple con Q-Learning por colores

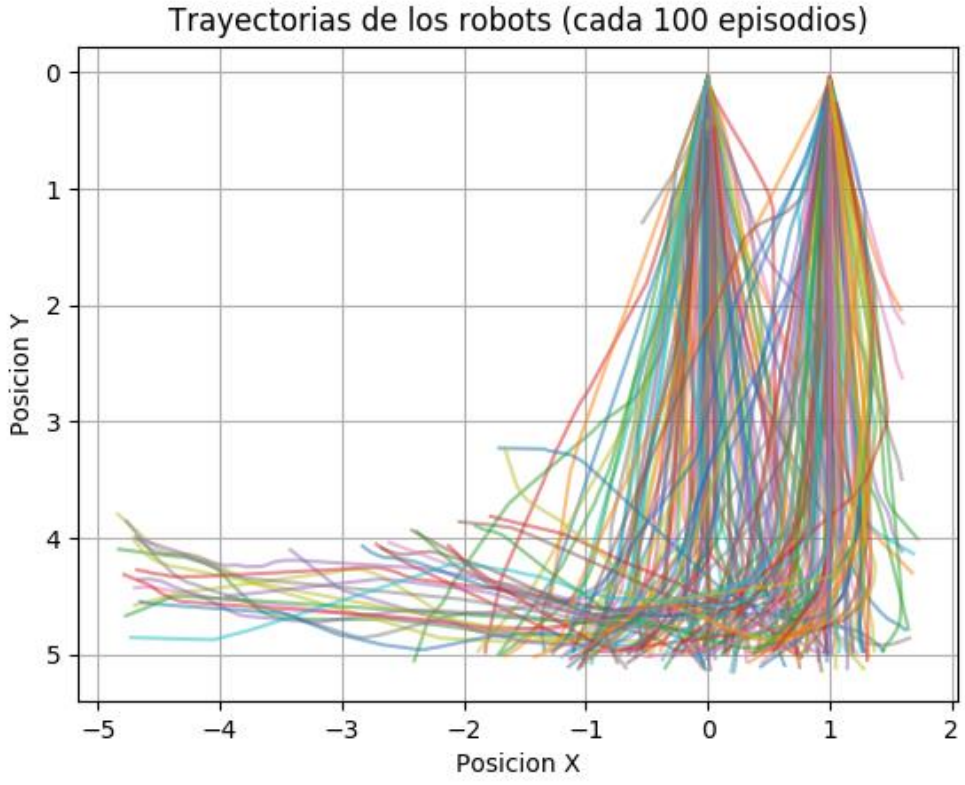


Figura 6-46. Trayectorias de los robots en uno de cada mil episodios del entrenamiento múltiple con Q-Learning

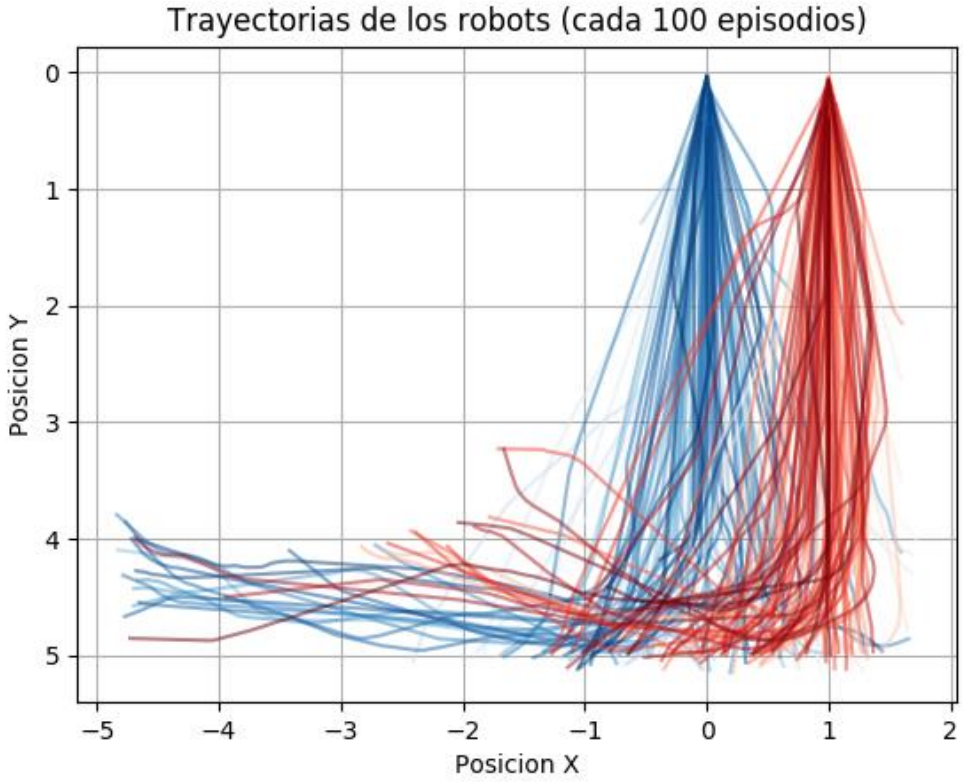


Figura 6-47. Trayectorias de los robots en uno de cada mil episodios del entrenamiento múltiple con Q-Learning por colores

Se han incluido estas cuatro figuras adicionales para ofrecer una perspectiva más completa sobre el progreso del aprendizaje del robot a través del tiempo. Juntas, estas representaciones muestran tanto el proceso de ajuste continuo como la consolidación de comportamientos óptimos del robot.

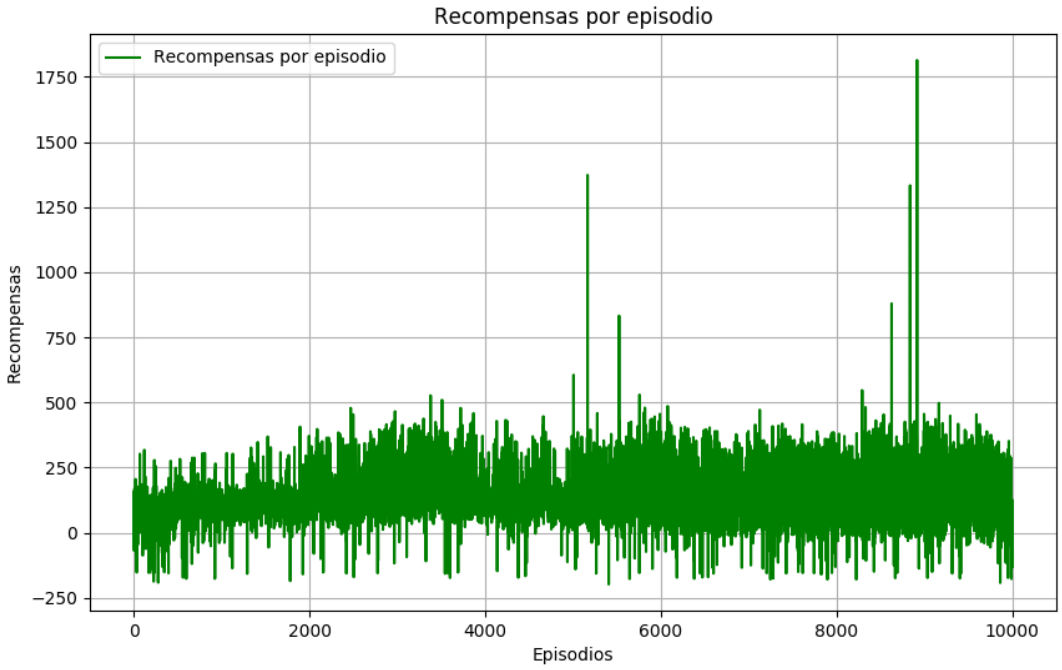


Figura 6-48. Recompensas por episodio en el primer entrenamiento multi-robot con Q-Learning

La figura 6-48 muestra una alta variabilidad en las recompensas por episodio en el caso multi-robot, con picos significativos alrededor de los episodios 5.000 y 9.000, donde las recompensas alcanzan valores extremadamente altos. A lo largo de los episodios, la mayoría de las recompensas se sitúan en el rango de -100 a 400, lo que indica una tendencia general hacia recompensas moderadas con algunos episodios de desempeño excepcionalmente alto. Este comportamiento sugiere que el entorno en el que operan los múltiples robots es complejo y dinámico, lo que provoca que las interacciones entre ellos resulten en un rendimiento muy variable. Los picos observados reflejan momentos en los que los robots lograron una navegación efectiva, permitiéndoles maximizar las recompensas.

Por otra parte, recordando la gráfica correspondiente al entrenamiento de un solo robot con estos mismos parámetros, se observaba que esta mostraba una variabilidad considerablemente menor en las recompensas, que eran más consistentes y rara vez superaban los 1.000. Esto implicaba un entorno más controlado y predecible, sugiriendo que el robot individual podía seguir una estrategia más estable y menos afectada por variables externas.

Comparando ambos escenarios, se observa que el entrenamiento multi-robot presenta más fluctuaciones y picos extremos en las recompensas. Esto evidencia que la competencia entre múltiples robots puede llevar a resultados muy diversos y menos predecibles. En contraste, el entrenamiento de un único robot resultaba en un desempeño más estable y predecible, con menos variabilidad en las recompensas. Así, mientras que el entorno multi-robot ofrece la posibilidad de alcanzar recompensas extremadamente altas, también introduce un nivel de complejidad y variabilidad que no se observa en el entrenamiento de un solo robot. Esta comparación pone de manifiesto las diferencias fundamentales en los desafíos y oportunidades que presentan ambos enfoques de entrenamiento con Q-Learning.

Figura 6-49. Histograma de recompensas por episodio en el primer entrenamiento multi-robot con Q-Learning

En primer lugar, al analizar la figura 6-49 observamos que la mayoría de las recompensas se agrupan entre los valores de 0 y 200. Este agrupamiento indica que, en la mayoría de los episodios, los robots logran obtener recompensas positivas, pero no excepcionalmente altas. Sin embargo, una característica notable de esta distribución es la presencia de una cola larga hacia la derecha que no es siquiera apreciable. Esta cola larga sugiere que, aunque no es frecuente, hay episodios en los que los robots alcanzan recompensas muy altas. Estos picos en las recompensas pueden ser indicativos de episodios en los que la navegación de los robots fue particularmente efectiva, permitiendo maximizar las ganancias de manera notable.

Además, esta distribución con una cola larga refleja una variabilidad considerable en las recompensas obtenidas. Es decir, mientras que en la mayoría de los episodios las recompensas se mantienen dentro de un rango

moderado, existen episodios ocasionales donde las recompensas se disparan. Esto puede deberse a la complejidad y la naturaleza dinámica del entorno multi-robot, donde las interacciones entre múltiples agentes pueden producir resultados impredecibles y, a veces, excepcionalmente buenos.

Por otro lado, el histograma de recompensas por episodio para un único robot mostraba una distribución bimodal, con picos en 0 y 300. Esto podía indicar que el robot seguía estrategias consistentes y efectivas, logrando recompensas predecibles y controladas. A diferencia del caso multi-robot, las recompensas extremadamente altas eran raras, lo que reflejaba un entorno más estable y menos variable.

Al comparar ambos casos, el entrenamiento del único robot mostraba una mayor consistencia en las recompensas, siguiendo estrategias efectivas y predecibles. En contraste, el multi-robot presenta una distribución más dispersa y una cola larga hacia la derecha, indicando mayor exploración y variabilidad debido a la complejidad de controlar múltiples robots. En resumen, el único robot es más constante y optimizado, mientras que el multi-robot refleja mayor exploración y variabilidad en las recompensas.

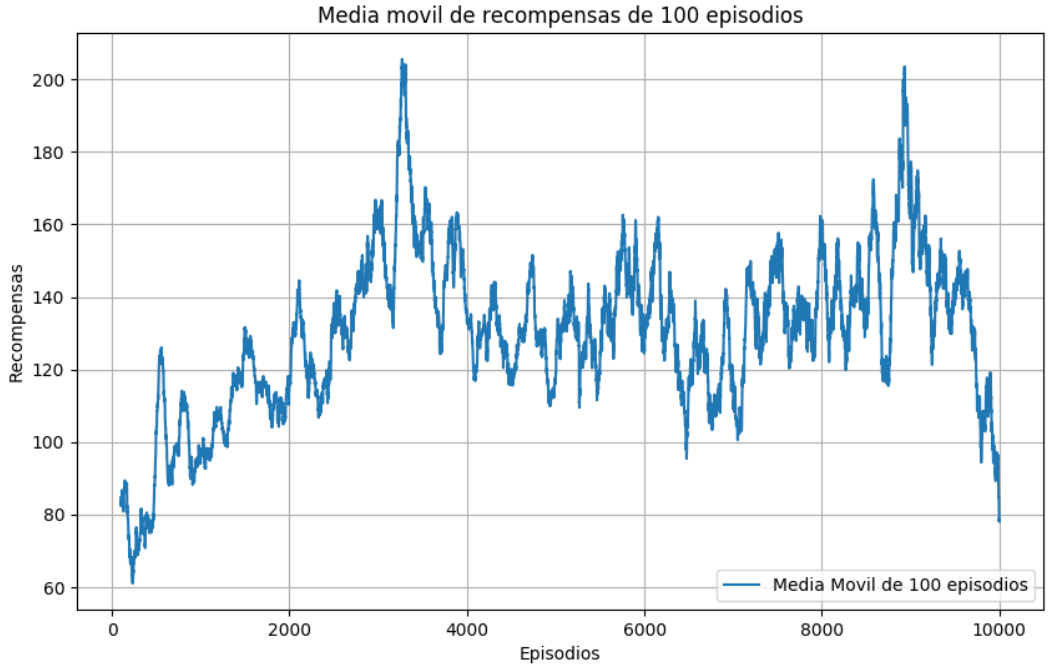


Figura 6-50. Medias móviles de las recompensas en el primer entrenamiento multi-robot con Q-Learning

En el caso de la media móvil de las recompensas, en la figura 6-50 se revela una tendencia ascendente general, lo que indica que los robots mejoran su rendimiento a lo largo del tiempo. Esta tendencia es una señal positiva, ya que demuestra que los robots están aprendiendo y adaptándose progresivamente para maximizar sus recompensas. Sin embargo, también se observan grandes fluctuaciones, especialmente hacia el final del entrenamiento. Estas fluctuaciones pueden ser atribuibles a la naturaleza dinámica y compleja de la interacción entre múltiples robots. Este patrón sugiere que, aunque los robots están mejorando en promedio, la navegación perfecta aún no se ha alcanzado de manera consistente, lo que provoca picos y caídas en la media móvil.

Por otro lado, el entrenamiento de un único robot también mostraba una tendencia ascendente en la media móvil de las recompensas, pero con menos fluctuaciones comparadas con el caso multi-robot, así como valores promedios mucho más elevados. Las recompensas promedio se estabilizaban hacia el final del entrenamiento, indicando que el robot había alcanzado un comportamiento más constante y optimizado. La ausencia de fluctuaciones significativas en el caso del único robot podía ser explicada por no tener que navegar con otros agentes. Sin la complejidad añadida de la competencia con otros robots, el único robot podía enfocarse en explorar y explotar estrategias que maximizaran sus recompensas de manera más directa y predecible.

Al comparar ambos escenarios, se puede observar que el entrenamiento multi-robot muestra más fluctuaciones en la media móvil de las recompensas. Esto es comprensible dado que controlar múltiples agentes en un entorno compartido es más complejo y susceptible a variaciones en el rendimiento individual y colectivo. La necesidad

de sincronización y la posibilidad de conflictos o interferencias entre robots contribuyen a estas fluctuaciones. En contraste, el entrenamiento de un único robot presentaba una mejora más constante y predecible. Sin la necesidad de compartir la navegación con otros robots, el único robot podía enfocarse en optimizar su propia estrategia de manera más eficiente, lo que resultaba en una curva de aprendizaje más suave y estable.

6.3.1.2. Entrenamiento 2

En la segunda simulación se hace uso de los parámetros que confieren el peor rendimiento al aprendizaje por refuerzo. Por ello, en el segundo entrenamiento realizado para entrenar a un sistema de dos robots utilizando el algoritmo Q-Learning, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,9	0,1	0,9	0,999

Tabla 6-11. Parámetros del algoritmo Q-Learning utilizados en el segundo entrenamiento multi-robot

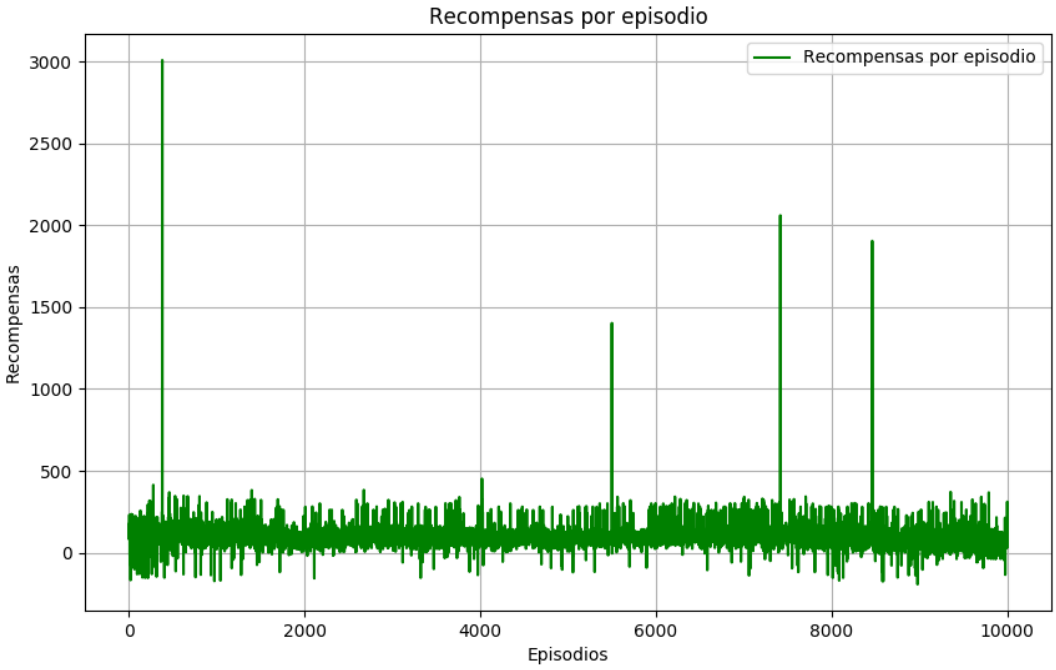


Figura 6-51. Recompensas por episodio en el segundo entrenamiento multi-robot con Q-Learning

La figura 6-51 muestra una variabilidad considerable en las recompensas obtenidas, una característica que también se observó en el entrenamiento multi-robot con diferentes parámetros. Se destacan picos muy altos al principio del entrenamiento y en episodios específicos a lo largo del proceso. Esto indica que en ciertos momentos los robots logran alcanzar recompensas mucho mayores. No obstante, en general las recompensas se mantienen en un rango menor cuando se comparan con los picos más altos, lo que sugiere que estos eventos de alta recompensa son excepcionales en lugar de ser la norma.

El entrenamiento multi-robot con un alto valor de alfa y un bajo valor de gamma implica que los robots están priorizando el aprendizaje de recompensas inmediatas en lugar de considerar las recompensas futuras. Este enfoque puede explicar los picos observados en la gráfica, ya que los robots podrían estar explotando estrategias que proporcionan recompensas altas de manera rápida. Sin embargo, estas estrategias no necesariamente son consistentes a largo plazo.

Como conclusión, aunque los parámetros actuales permiten a los robots alcanzar altos picos de recompensas, la

falta de consideración de las recompensas futuras y la dificultad en la navegación conjunta resultan en una alta variabilidad y una falta de consistencia en el rendimiento global.

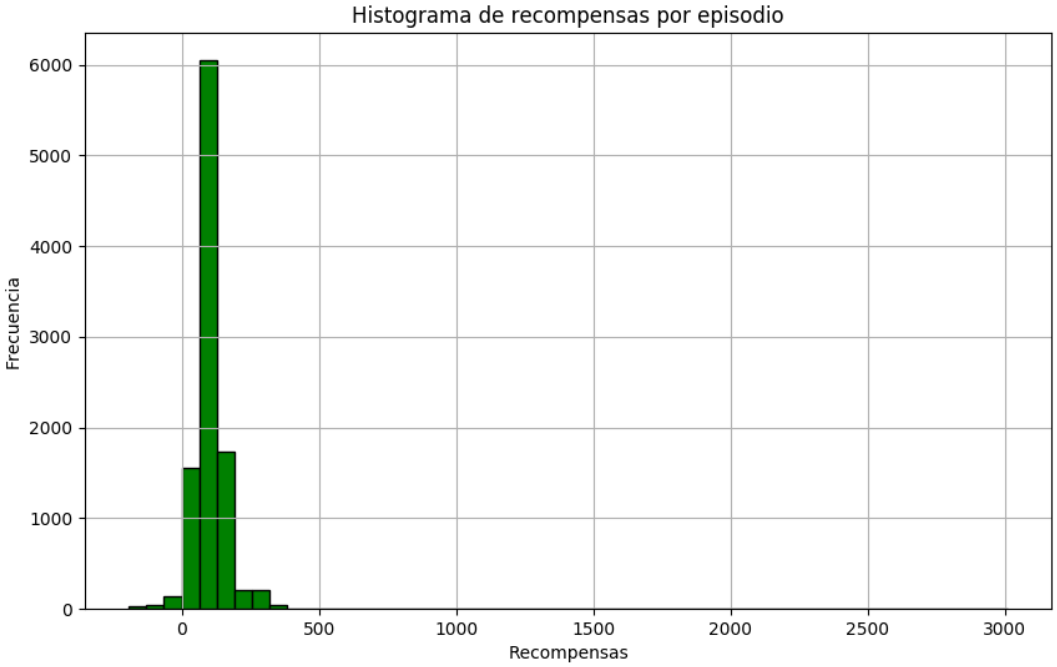


Figura 6-52. Histograma de recompensas por episodio en el primer entrenamiento multi-robot con Q-Learning

La figura 6-52 revela una gran concentración de episodios con recompensas situadas entre 0 y 100, lo que indica que la mayoría de los episodios no obtienen recompensas significativamente altas. Esta distribución tan estrecha sugiere que, aunque los robots logran obtener recompensas en la mayoría de los episodios, estas recompensas son, en su mayoría, bastante modestas. Se aprecia fácilmente que hay muy pocos episodios que alcanzan recompensas extremadamente altas, lo que implica que estas ocurrencias son excepcionales y no representan la norma en el rendimiento de los robots.

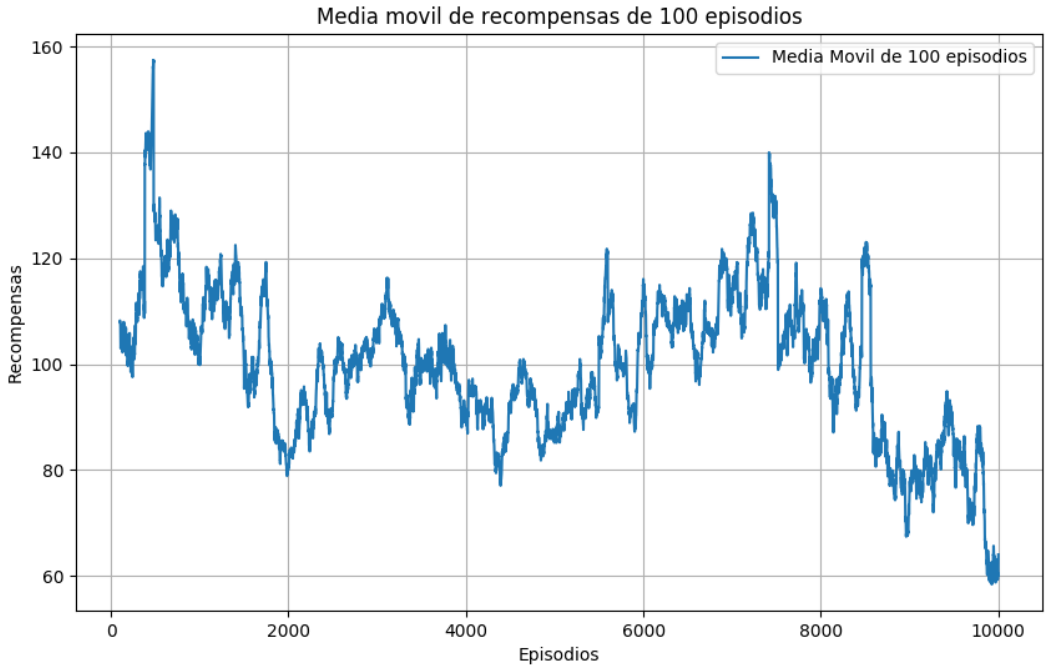


Figura 6-53. Medias móviles de las recompensas en el segundo entrenamiento multi-robot con Q-Learning

La gráfica de la media móvil de recompensas revela una tendencia decreciente a lo largo del tiempo. Inicialmente, se puede observar un pico significativo en las recompensas, lo que indica que los robots logran encontrar estrategias efectivas y beneficiosas al inicio del entrenamiento. Sin embargo, conforme avanzan los episodios, la media móvil de las recompensas comienza a descender gradualmente y tiende a estabilizarse en un valor mucho más bajo. Este comportamiento indica que las recompensas iniciales obtenidas no son sostenibles a lo largo del tiempo y que los robots encuentran dificultades para mantener un rendimiento elevado de manera constante.

El patrón decreciente en la media móvil de recompensas indica que, aunque los robots encuentran y explotan estrategias efectivas al inicio del entrenamiento, estas no son sostenibles a largo plazo debido al alto valor de alfa, que fomenta un rápido aprendizaje de recompensas inmediatas, y al bajo gamma que reduce la consideración de recompensas futuras. Esta combinación lleva a una explotación rápida de estrategias a corto plazo, resultando en una disminución de la eficiencia general del sistema conforme avanza el entrenamiento.

No obstante, al comparar este caso con el entrenamiento de un solo robot bajo las mismas condiciones (mismo algoritmo y parámetros), se observa que las recompensas obtenidas en el entorno multi-robot son más altas que en el caso de un único robot.

El hecho de que, en un caso concreto, el rendimiento del entorno multi-robot con una tasa de aprendizaje alta y un factor de descuento bajo (alfa 0,9 y gamma 0,1) supere al entorno de un solo robot, mientras que con una tasa de aprendizaje baja y un factor de descuento alto (alfa 0,1 y gamma 0,8), el rendimiento del multi-robot sea inferior, puede justificarse mediante el análisis de cómo estos parámetros afectan el aprendizaje y la interacción entre los robots.

En el caso de alfa 0,9 y gamma 0,1, el entorno multi-robot se beneficia de una alta tasa de aprendizaje porque permite a los robots ajustarse rápidamente a los cambios en el entorno, lo cual es beneficioso en un entorno dinámico y complejo donde la interacción entre robots puede causar variaciones frecuentes. Además, un bajo gamma implica que los robots se enfocan más en recompensas inmediatas. En un entorno multi-robot, esta estrategia puede reducir la complejidad de la navegación con obstáculos móviles a largo plazo, permitiendo que los robots se adapten rápidamente a las recompensas actuales sin tener que considerar interacciones futuras complejas.

Por otro lado, aunque un alto alfa permite un aprendizaje rápido en un entorno de un solo robot, la estabilidad

de este entorno hace que no se necesiten ajustes tan rápidos. La diferencia en rendimiento no es tan marcada como en el entorno multi-robot, y un bajo gamma en un entorno de un solo robot resulta no ser tan beneficioso, ya que no hay interacciones complejas que gestionar, y las estrategias a largo plazo resultan ser más efectivas.

En el caso de alfa 0,1 y gamma 0,8, un bajo alfa implica que los robots actualizan sus estrategias lentamente, lo que resulta ser problemático en un entorno dinámico con interacciones complejas. La lenta adaptación lleva a un desempeño inferior en el entorno multi-robot. Además, un alto gamma significa que los robots se enfocan en recompensas futuras. Dado que la navegación en entornos multi-robot ya es difícil, este enfoque puede resultar en un rendimiento inconsistente y menor.

Sin embargo, un bajo alfa es menos problemático en un entorno estable de un solo robot, permitiendo un aprendizaje más gradual y estable. Un alto gamma es beneficioso en este caso, ya que el robot puede planificar a largo plazo sin la necesidad de considerar interacciones complejas, optimizando así sus estrategias de manera más efectiva.

Como conclusión, el entorno multi-robot se beneficia de una alta tasa de aprendizaje y un bajo factor de descuento (alfa 0,9 y gamma 0,1) porque permite adaptaciones rápidas a las recompensas inmediatas y reduce la complejidad de la navegación con obstáculos móviles a largo plazo. Por otro lado, el entorno de un solo robot no se beneficia tanto de una rápida adaptación porque es más estable, y un enfoque a largo plazo es más adecuado. Con una baja tasa de aprendizaje y un alto enfoque en el futuro (alfa 0,1 y gamma 0,8) el entorno multi-robot sufre debido a la complejidad, mientras que el entorno de un solo robot puede seguir optimizando de manera efectiva. Estas diferencias subrayan la importancia de ajustar los parámetros de aprendizaje según el entorno específico y las interacciones que predominan en él.

6.3.1.3. Entrenamiento 3

En la tercera simulación multi-robot empleando el algoritmo Q-Learning se utilizará de nuevo la combinación más favorable en el entrenamiento. Por ello, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,1	0,8	0,9	0,999

Tabla 6-12. Parámetros del algoritmo Q-Learning utilizados en el tercer entrenamiento multi-robot

En este caso, el entrenamiento se realizará en un entorno más amplio para analizar cómo la restricción de espacio afecta al rendimiento de los robots. El obstáculo central no conforma un estrecho recorrido como en los casos anteriores, sino que simplemente se trata de un cuadrado que los robots deben bordear para llegar al punto de partida y completar el laberinto.

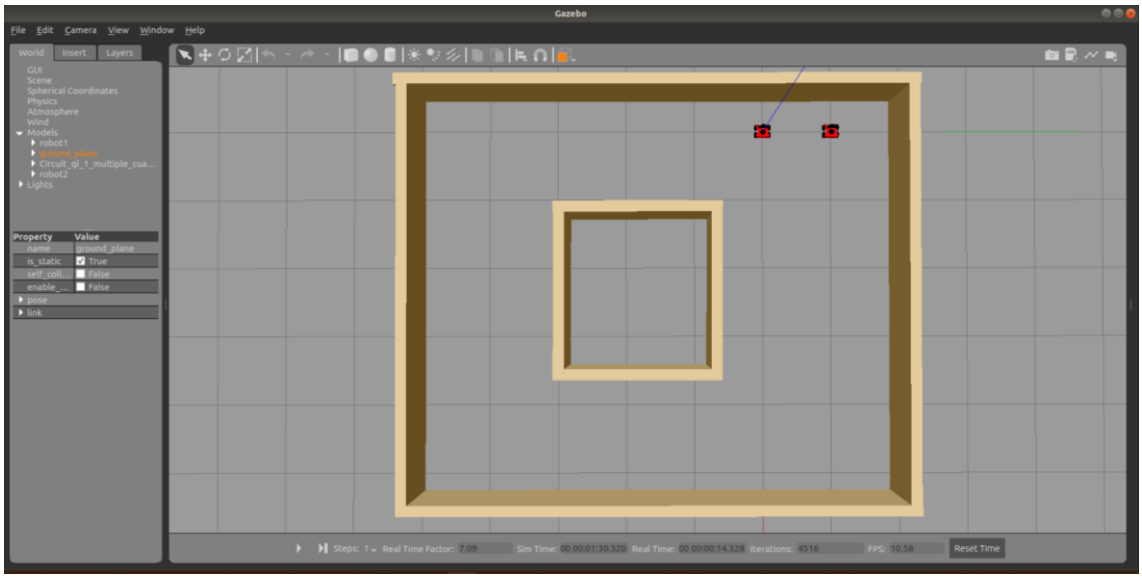


Figura 6-54. Entorno de simulación multi-robot amplio

Las trayectorias seguidas por los dos robots en cada uno de los episodios se muestran a continuación, con el objetivo de poder visualizar el comportamiento de los agentes antes de pasar a analizar las recompensas obtenidas.

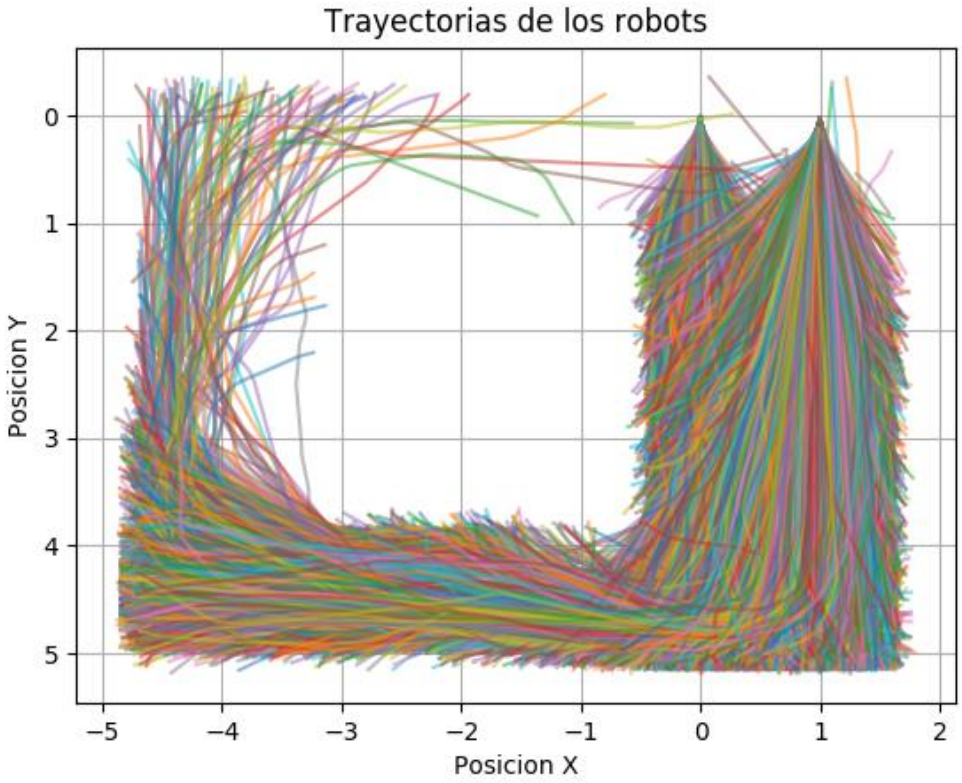


Figura 6-55. Trayectorias de los robots en cada uno de los episodios del entrenamiento múltiple ancho con Q-Learning

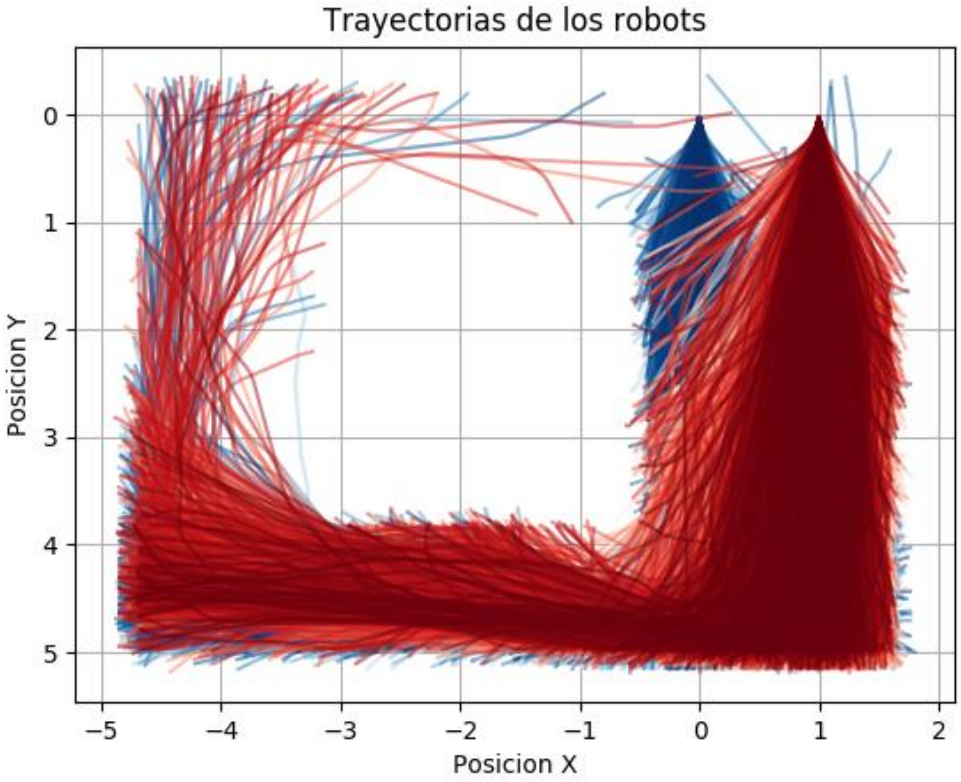


Figura 6-56. Trayectorias de los robots en cada episodio del entrenamiento múltiple ancho con Q-Learning por colores

En las figuras 6-55 y 6-56 se muestran las trayectorias llevadas a cabo por cada uno de los robots a lo largo del entrenamiento. En contraste con el escenario anterior, donde el entorno era un laberinto con espacio limitado, en este caso se observa que en varios episodios los robots logran llegar a la meta y completar el recorrido. Esto evidencia que el problema en el entorno anterior radicaba en la falta de espacio, lo cual impedía que los robots avanzaran adecuadamente debido a las políticas que estaban siguiendo. Sin embargo, la cantidad de episodios en los que los robots completan el recorrido es considerablemente menor que en el caso de un solo robot, principalmente debido a la mayor complejidad introducida por la presencia de un segundo robot como obstáculo móvil.

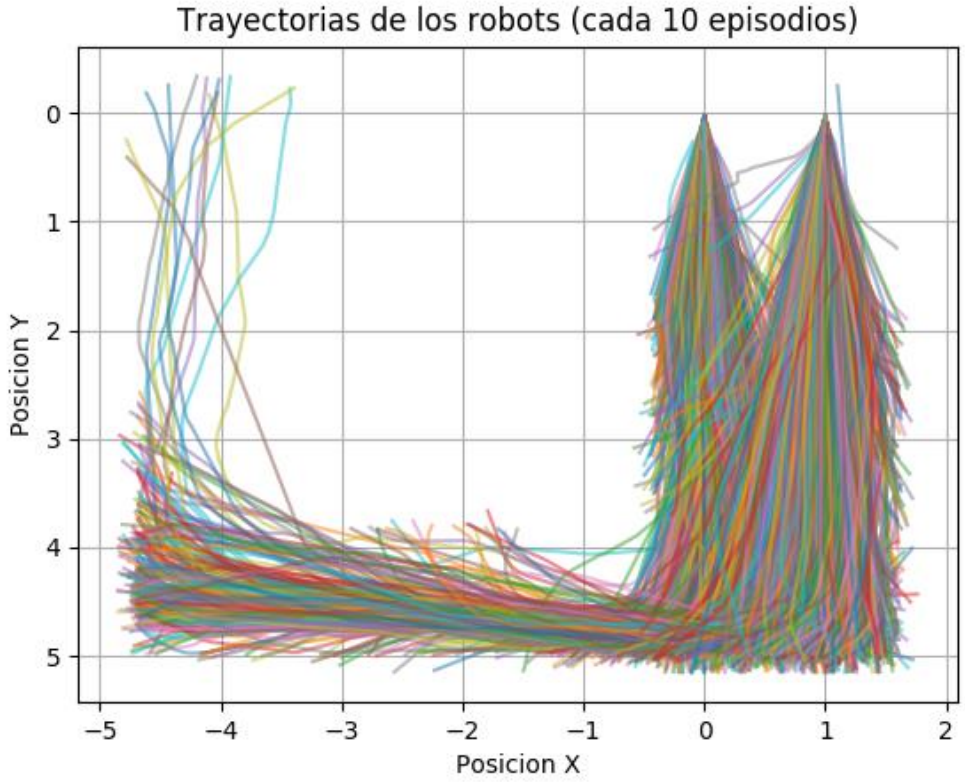


Figura 6-57. Trayectorias de los robots en uno de cada cien episodios del entrenamiento múltiple ancho con Q-Learning

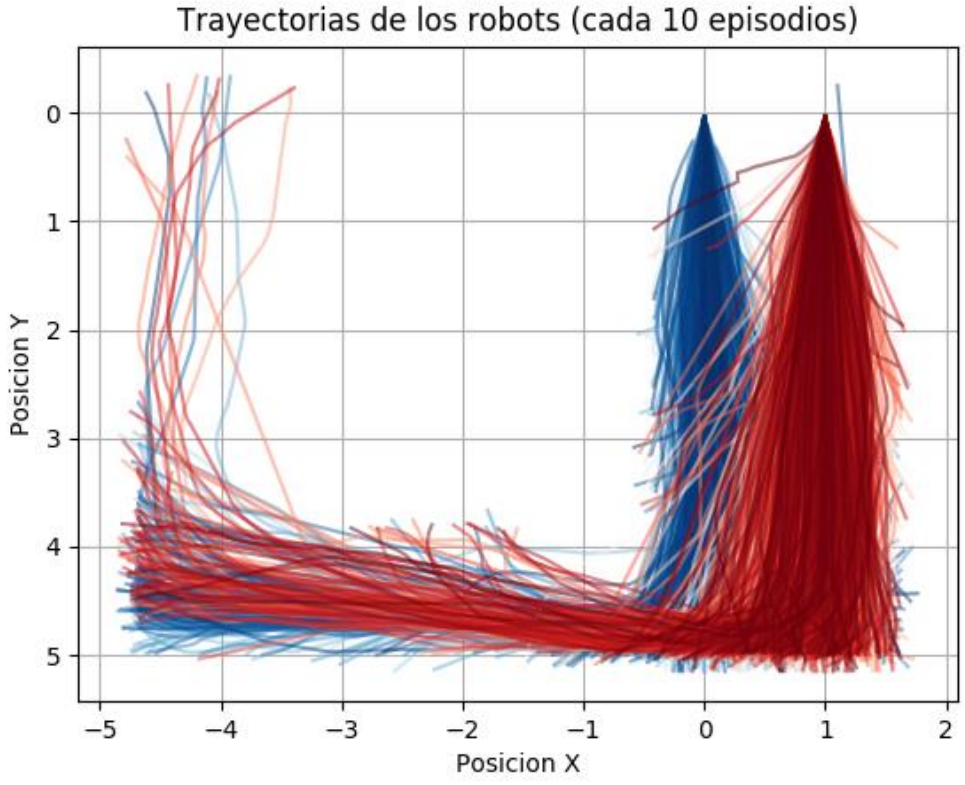


Figura 6-58. Trayectorias de los robots en uno de cada cien episodios del entrenamiento múltiple ancho con Q-Learning por colores

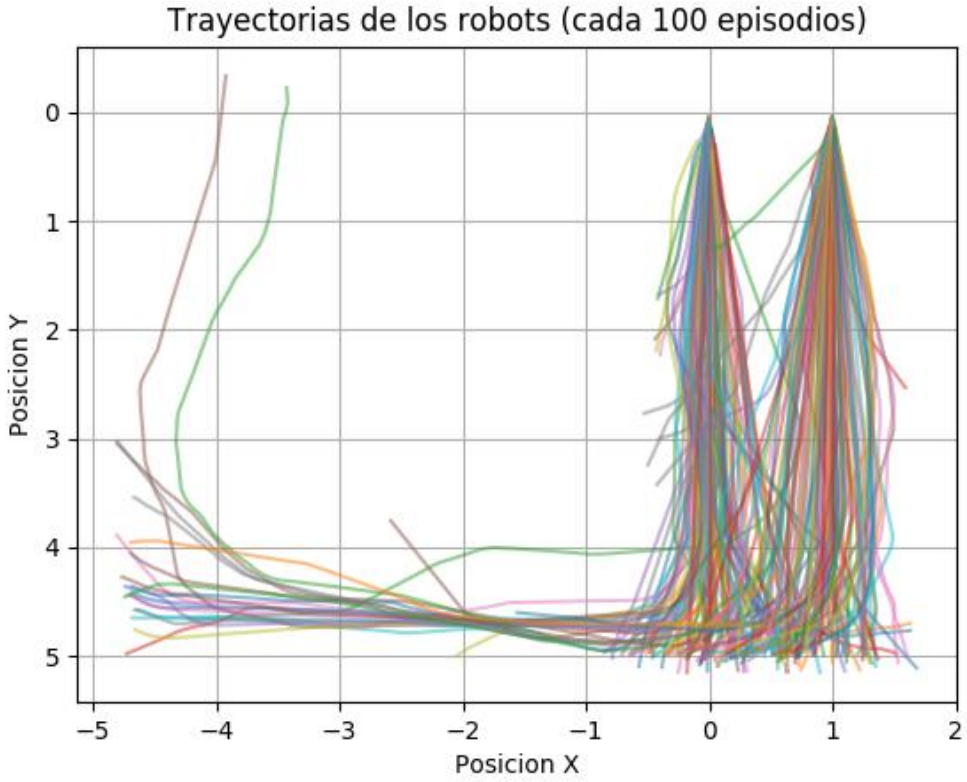


Figura 6-59. Trayectorias de los robots en uno de cada mil episodios del entrenamiento múltiple ancho con Q-Learning

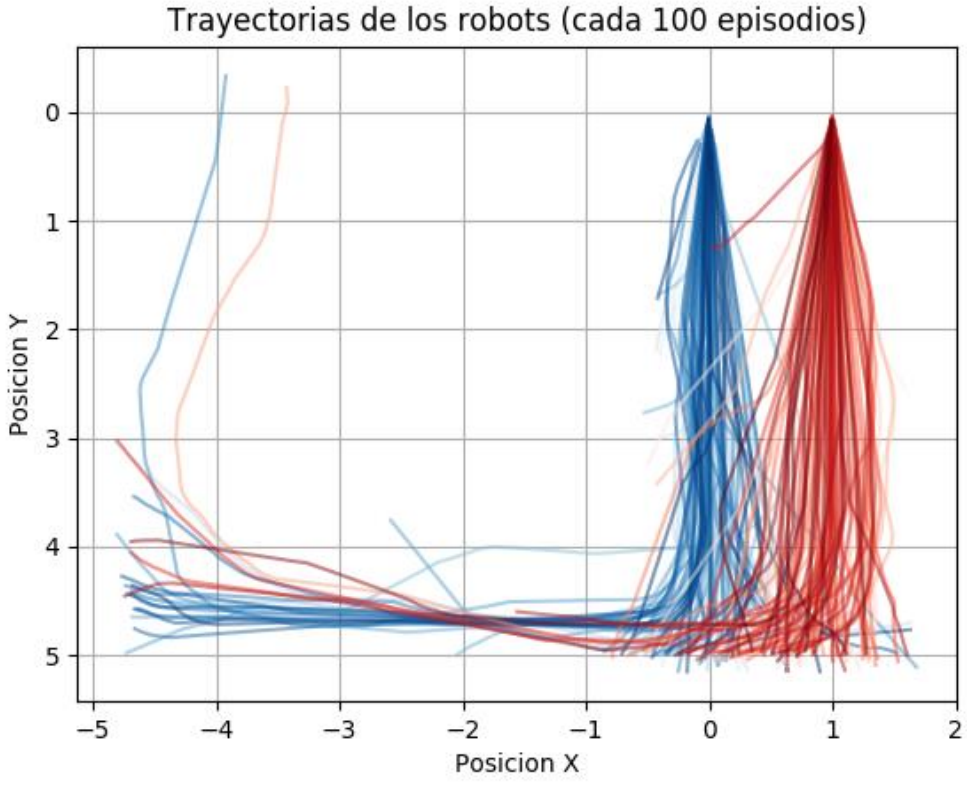


Figura 6-60. Trayectorias de los robots en uno de cada mil episodios del entrenamiento múltiple ancho con Q-

Learning por colores

Se han incluido estas cuatro figuras adicionales para ofrecer una perspectiva más completa sobre el progreso del aprendizaje del robot a través del tiempo. Juntas, estas representaciones muestran tanto el proceso de ajuste continuo como la consolidación de comportamientos óptimos del robot.

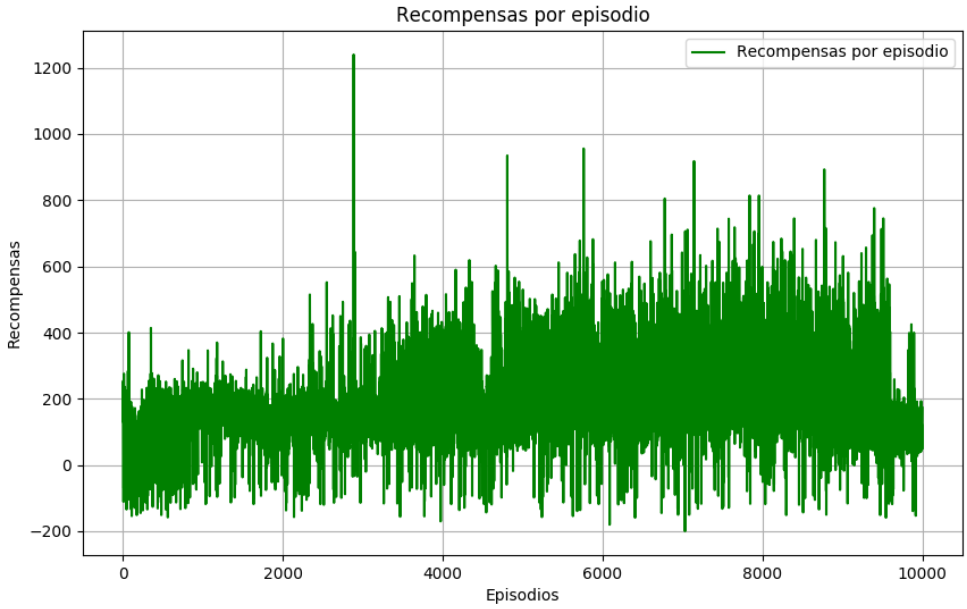


Figura 6-61. Recompensas por episodio en el tercer entrenamiento multi-robot con Q-Learning

La figura 6-61 revela una considerable variabilidad en las recompensas obtenidas por episodio en el escenario multi-robot, con picos notables en ciertos episodios a lo largo del entrenamiento, donde las recompensas alcanzan niveles extremadamente altos. A medida que avanzan los episodios, el rango de recompensas se expande gradualmente, lo que sugiere una tendencia general de mejora, con algunos episodios de desempeño excepcionalmente alto y otros más bajos. Este patrón indica que los múltiples robots operan en un entorno complejo y dinámico, lo que genera interacciones que resultan en un rendimiento altamente variable.

Por otro lado, al recordar la gráfica del entrenamiento en un entorno más reducido con los mismos parámetros, se observaba que las recompensas presentaban una variabilidad significativamente menor y también eran mucho más bajas, ya que raramente superaban los 400 puntos. Esto indicaba un entorno en el que los robots enfrentaban grandes dificultades para avanzar a partir de cierto punto, donde quedaban atrapados debido a la falta de espacio.

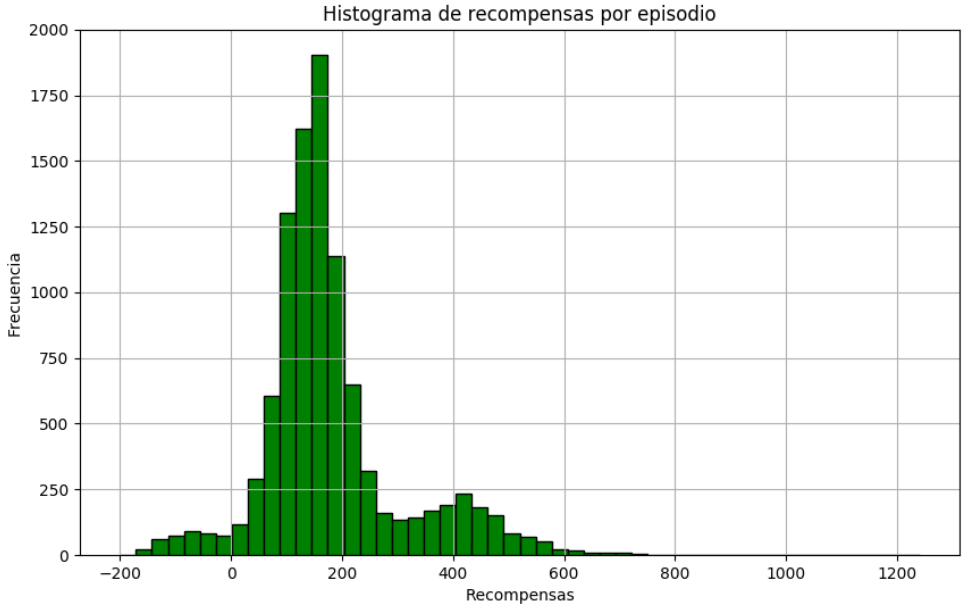


Figura 6-62. Histograma de recompensas por episodio en el tercer entrenamiento multi-robot con Q-Learning

Al analizar el histograma de recompensas en el entorno más amplio para el caso multi-robot, en la figura 6-62, se observa una extensión del rango de recompensas de -200 a 600, lo que indica una mayor variabilidad en el desempeño de los robots comparado con el entorno más estrecho. Esta variabilidad sugiere que el entorno amplio proporciona más libertad para el movimiento y la exploración de estrategias, permitiendo a los robots experimentar una gama más amplia de resultados.

El histograma muestra dos picos principales: uno alrededor de 150, con aproximadamente 1900 episodios, y otro cerca de 400, con alrededor de 250 episodios. Aunque la mayoría de las recompensas están en valores modestos, la presencia de un segundo pico en torno a 400 indica que en algunos episodios los robots logran maximizar sus recompensas, aprovechando las oportunidades del entorno amplio. En contraste con el entorno estrecho, donde las recompensas estaban más concentradas entre 0 y 200 y se observaba una cola larga hacia la derecha, el entorno amplio refleja una exploración más eficaz y patrones recurrentes de éxito en las recompensas.

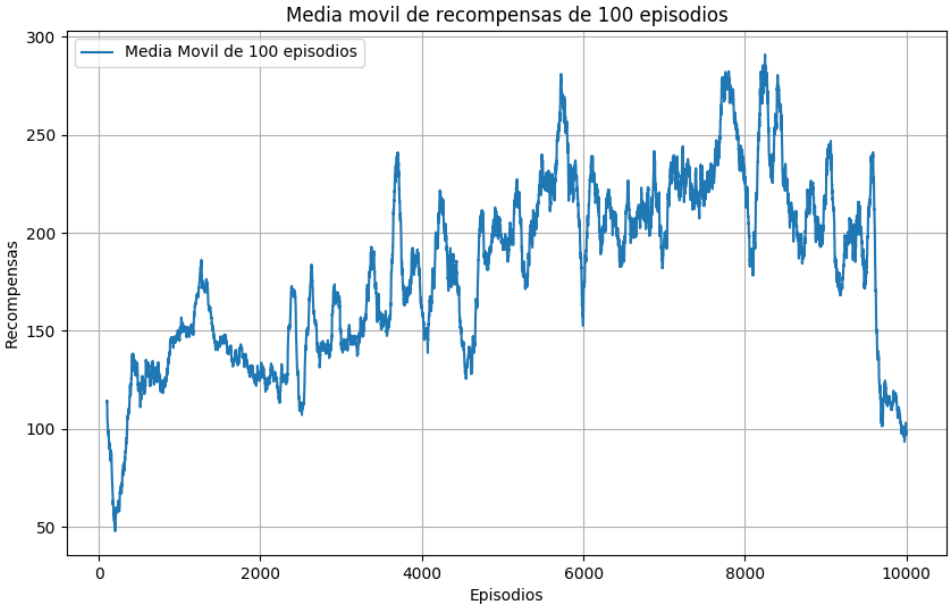


Figura 6-63. Medias móviles de las recompensas en el tercer entrenamiento multi-robot con Q-Learning

En comparación con el entorno estrecho, el nuevo entorno más amplio para el entrenamiento multi-robot muestra una tendencia notablemente diferente en las medias móviles de las recompensas, fácilmente observable en la figura 6-63. En el entorno estrecho, las medias móviles exhibían una mejora bastante limitada, estabilizándose en valores modestos con algunas fluctuaciones hacia el final. Esto indicaba que, aunque los robots mejoraban, las restricciones del entorno limitaban la exploración y la optimización, alcanzando una cierta meseta en el rendimiento.

En contraste, en el entorno más amplio, las medias móviles de las recompensas aumentan progresivamente hasta alcanzar un pico de aproximadamente 300 en el episodio 8000. Esta tendencia ascendente sostenida sugiere que el entorno más amplio facilita una mayor exploración y optimización de estrategias, permitiendo a los robots mejorar de manera más consistente y significativa. La mayor libertad y espacio en el entorno amplio contribuyen a un rendimiento más alto y estable, en comparación con el entorno más restringido.

6.3.2. Sarsa

6.3.2.1. Entrenamiento 1

En la primera simulación multi-robot empleando el algoritmo Sarsa se utilizará de nuevo la combinación más favorable para el entrenamiento. Por ello, en el primer entrenamiento realizado para entrenar a un sistema de dos robots utilizando el algoritmo Sarsa, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,1	0,8	0,9	0,999

Tabla 6-13. Parámetros del algoritmo Sarsa utilizados en el primer entrenamiento multi-robot

Las trayectorias seguidas por los dos robots en cada uno de los episodios se muestran a continuación, con el objetivo de poder visualizar el comportamiento de los agentes antes de pasar a analizar las recompensas obtenidas.

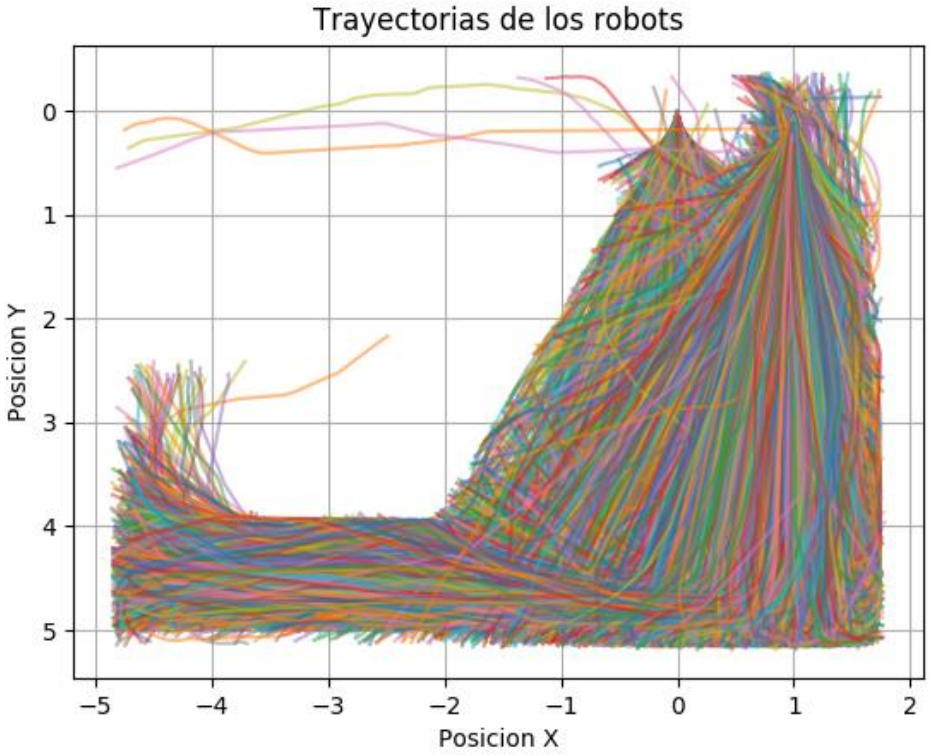


Figura 6-64. Trayectorias de los robots en cada episodio del entrenamiento múltiple con Sarsa

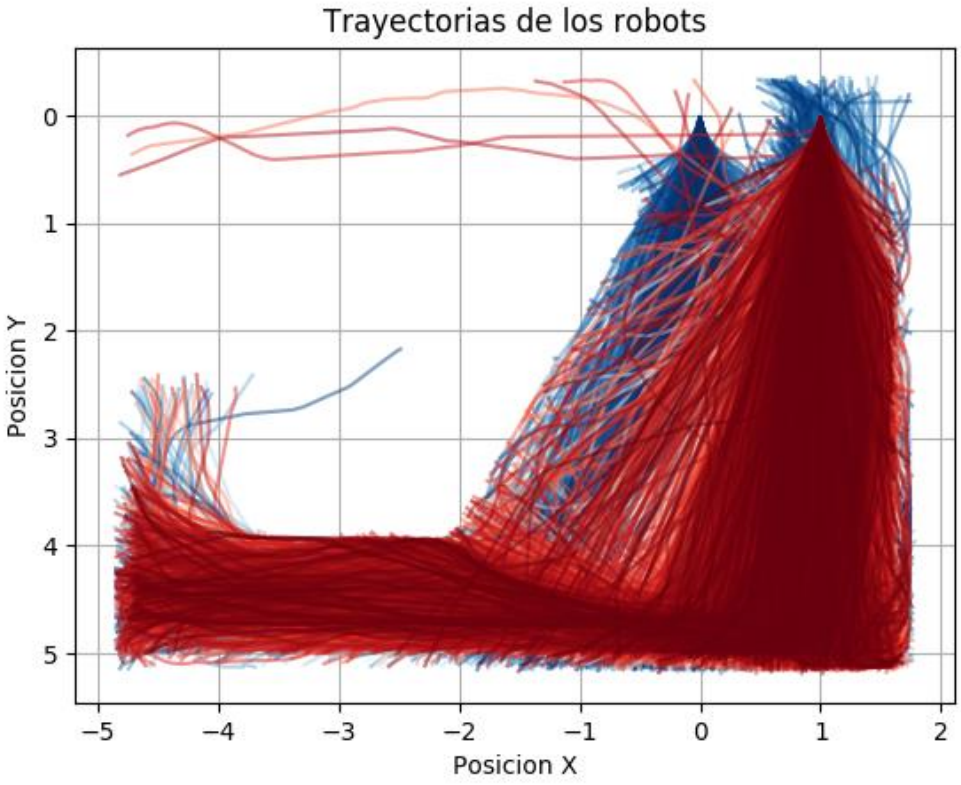


Figura 6-65. Trayectorias de los robots en uno de cada cien episodios del entrenamiento múltiple con Sarsa por colores

Las figuras 6-64 y 6-65 muestran las trayectorias de dos robots durante el entrenamiento, revelando una mayor

complejidad en las rutas exploradas en comparación con el caso de un solo robot. Ninguno de los dos logra completar el laberinto, probablemente debido a la falta de espacio en la zona donde sus trayectorias se detienen, lo que provoca que se bloqueen entre sí o con las paredes, al igual que ocurría en el caso del algoritmo Q-Learning.

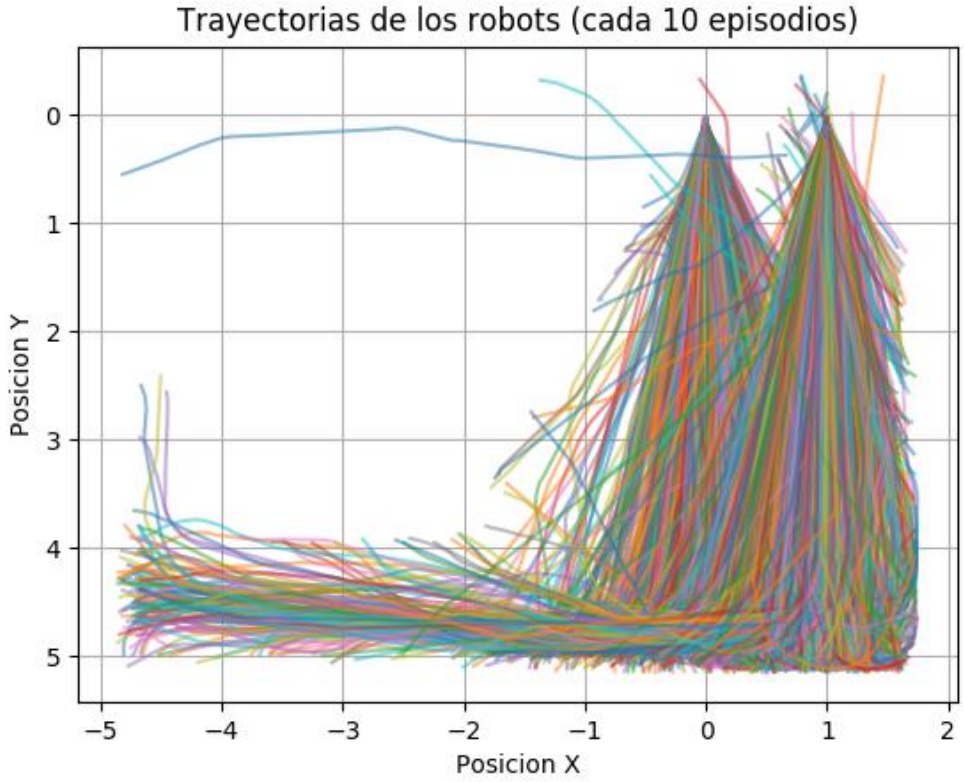


Figura 6-66. Trayectorias de los robots en uno de cada cien episodios del entrenamiento múltiple con Sarsa

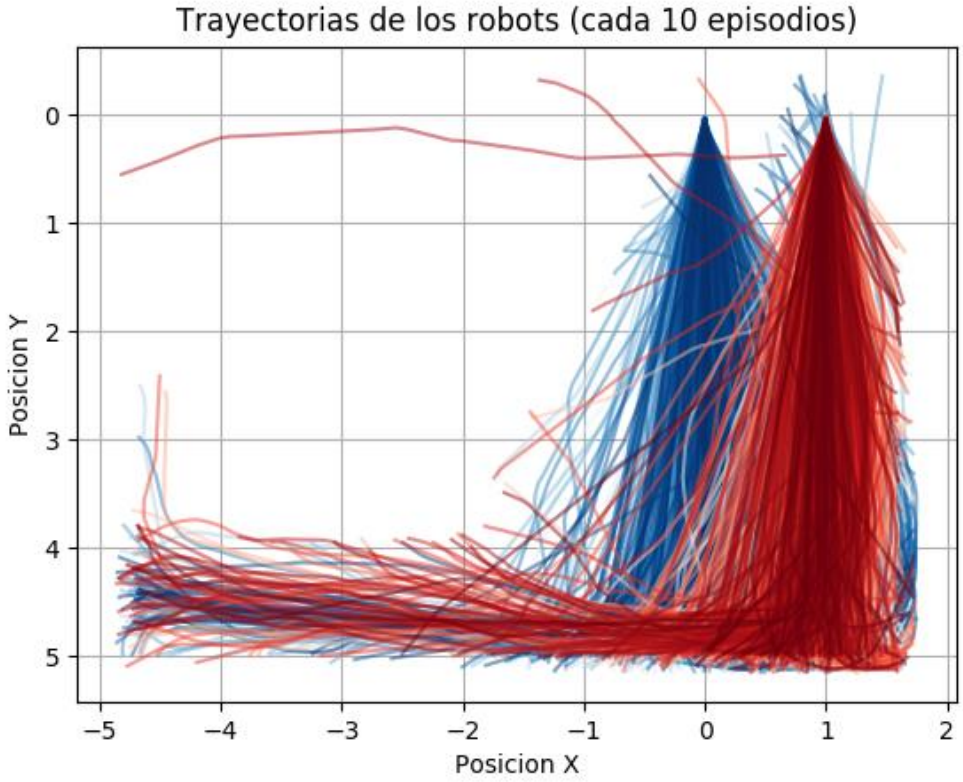


Figura 6-67. Trayectorias de los robots en uno de cada cien episodios del entrenamiento múltiple con Sarsa por colores

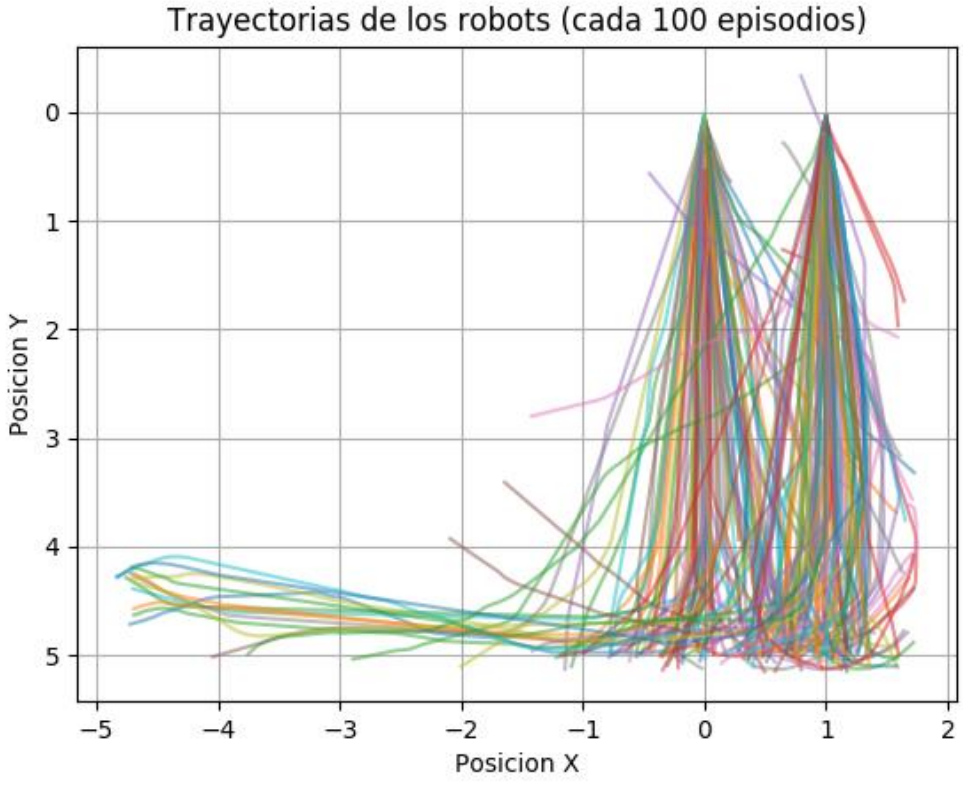


Figura 6-68. Trayectorias de los robots en uno de cada mil episodios del entrenamiento múltiple con Sarsa

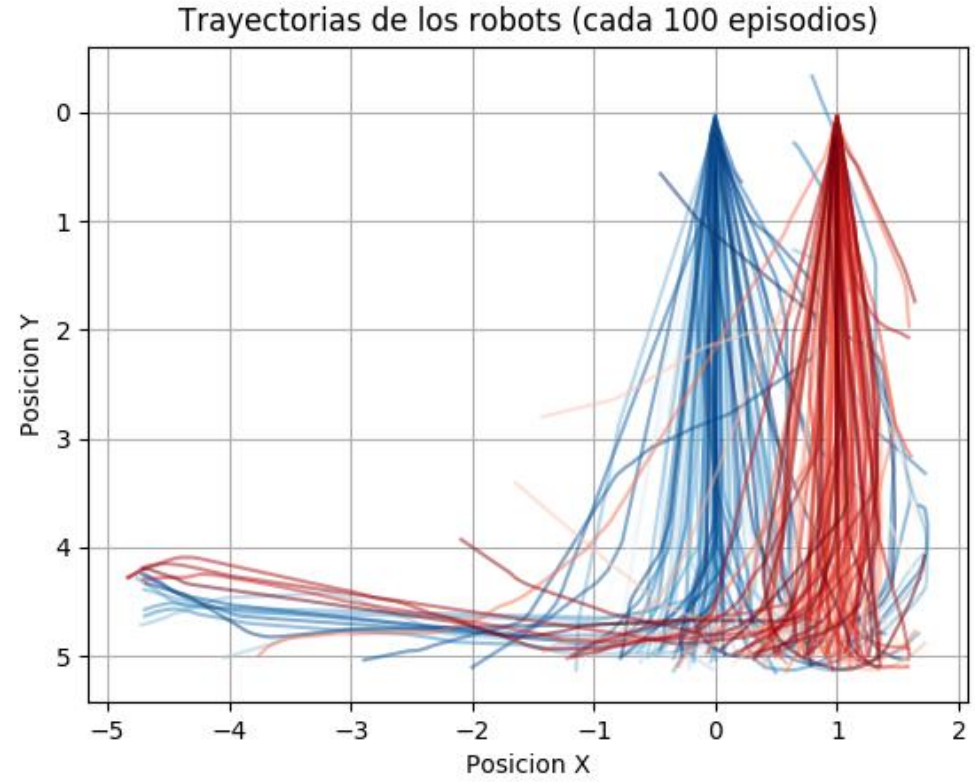


Figura 6-69. Trayectorias de los robots en uno de cada mil episodios del entrenamiento múltiple con Sarsa por colores

Se han incluido estas cuatro figuras adicionales para ofrecer una perspectiva más completa sobre el progreso del aprendizaje del robot a través del tiempo. Juntas, estas representaciones muestran tanto el proceso de ajuste continuo como la consolidación de comportamientos óptimos del robot.

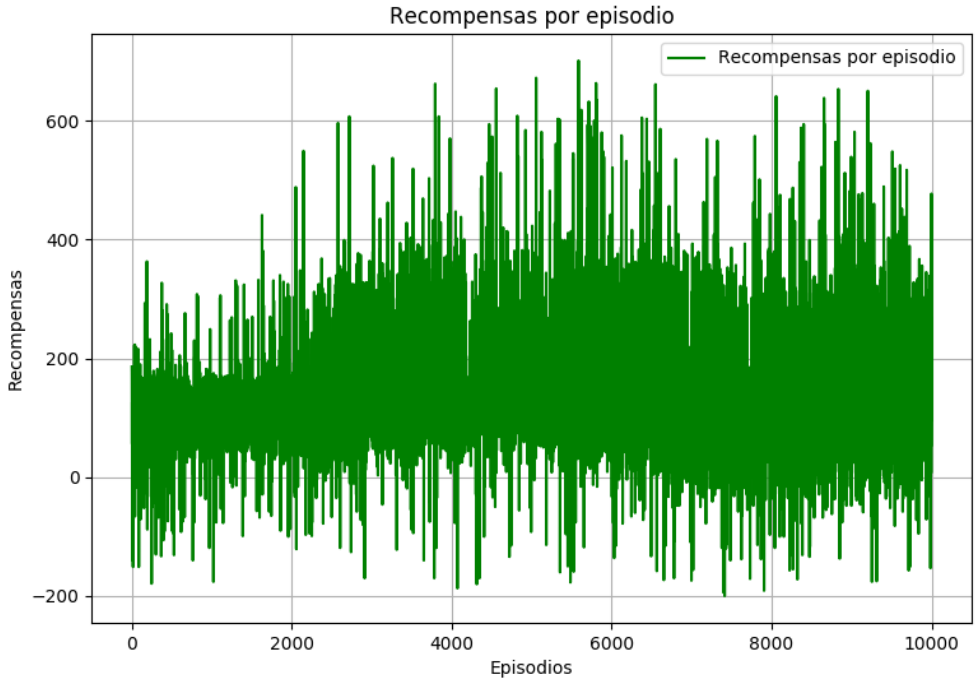


Figura 6-70. Recompensas por episodio en el primer entrenamiento multi-robot con Sarsa

En la gráfica de recompensas por episodio utilizando el algoritmo Sarsa, representada en la figura 6-70, se observa una considerable variabilidad, caracterizada por un aumento progresivo en la amplitud de las recompensas a medida que avanza el tiempo de entrenamiento. Aunque existen picos ocasionales que destacan por su altura, las recompensas tienden a mantenerse dentro de un rango más controlado, sin desviaciones extremas frecuentes. Esta variabilidad, vista a través de la perspectiva de aprendizaje de los robots, sugiere que están mejorando continuamente sus estrategias para maximizar recompensas a mediano y largo plazo.

Por otro lado, la gráfica de recompensas por episodio con el algoritmo Q-Learning mostraba una mayor variabilidad en comparación con Sarsa, con picos más pronunciados, aunque menos frecuentes. Las recompensas oscilaban de manera más dramática, lo que indicaba episodios donde las recompensas alcanzaban niveles muy altos, intercalados con otros episodios de rendimiento significativamente menor. Este patrón sugería que los robots con Q-Learning estaban explotando estrategias que podían generar recompensas muy altas en ciertos episodios específicos. Sin embargo, la menor frecuencia de estos picos y la mayor variabilidad observada en la gráfica indican que, aunque las recompensas pueden ser extremadamente altas, la consistencia en el rendimiento no es tan sólida como la observada con Sarsa.

Comparando ambos algoritmos, Sarsa presenta una distribución de recompensas más controlada y consistente a lo largo del tiempo, lo que se traduce en un rendimiento más predecible para los robots. Mientras tanto, Q-Learning permite alcanzar picos de recompensas más altos, pero con una mayor variabilidad, lo que implica que, aunque ofrece oportunidades para obtener recompensas extremas, estas no se producen de manera constante y predecible. Por lo tanto, Sarsa es más adecuado para situaciones donde la estabilidad y la consistencia del rendimiento son cruciales, mientras que Q-Learning puede ser preferible en escenarios donde la posibilidad de alcanzar recompensas muy altas es más valorada, aunque a costa de una menor predictibilidad en el rendimiento.

Figura 6-71. Histograma de recompensas por episodio en el primer entrenamiento multi-robot con Sarsa

La figura 6-71, correspondiente al histograma de recompensas del entrenamiento multi-robot utilizando el algoritmo Sarsa, muestra una notable concentración de episodios con recompensas situadas entre 50 y 200, lo que denota una distribución bastante uniforme. Este patrón sugiere que la mayoría de los episodios resultan en recompensas dentro de este rango, con pocos episodios alcanzando recompensas extremadamente altas. La uniformidad en la distribución indica que Sarsa permite a los robots tomar decisiones que maximizan las recompensas, mostrando una estabilidad destacable en las estrategias aprendidas. Esta consistencia sugiere que los robots están adoptando estrategias efectivas que se aplican de manera regular y confiable en diversas situaciones del entorno de entrenamiento.

En comparación, el histograma de recompensas por episodio del entrenamiento multi-robot utilizando Q-Learning revelaba una concentración similar de episodios en el rango de 100 a 200 recompensas, pero con una notable diferencia: la presencia de una cola más larga hacia recompensas más altas. Esta cola más extendida indicaba que, aunque la mayoría de las recompensas se encontraban dentro del rango de 100 a 200, había episodios en los que se alcanzaban recompensas significativamente más elevadas. Esta distribución sugiere que Q-Learning, aunque presenta menos consistencia en comparación con Sarsa, permite la posibilidad de alcanzar recompensas extremadamente altas en ciertos episodios.

Al comparar ambos histogramas, se observa que el histograma de Sarsa muestra una mayor estabilidad y uniformidad en las recompensas, lo que sugiere una estrategia de aprendizaje más consistente y predecible. En contraste, el histograma de Q-Learning, con su cola más larga hacia recompensas altas, indica una mayor variabilidad y la posibilidad de alcanzar recompensas extremas.

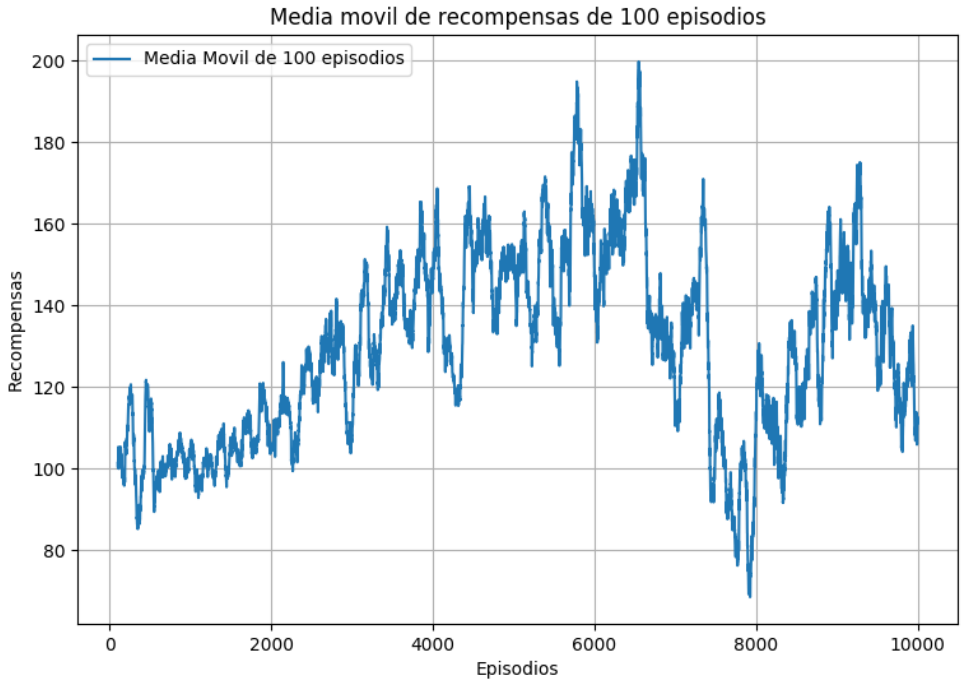


Figura 6-72. Medias móviles de las recompensas en el primer entrenamiento multi-robot con Sarsa

En el caso del entrenamiento multi-robot utilizando Sarsa, la media móvil de recompensas muestra una tendencia ascendente más estable, alcanzando picos cercanos a los 200. Sin embargo, lo más notable es la menor cantidad de fluctuaciones significativas en comparación con Q-Learning. Esta estabilidad en la tendencia ascendente sugiere que Sarsa permite a los robots una mejora continua y consistente en su rendimiento.

En contraste, la media móvil de recompensas para Q-Learning también mostraba una tendencia ascendente, alcanzando picos comparables a los observados con Sarsa, pero con fluctuaciones ligeramente más pronunciadas.

Estas diferencias reflejan cómo cada algoritmo maneja la exploración y explotación de estrategias: mientras Sarsa favorece una política que maximiza recompensas de manera estable y uniforme, Q-Learning permite una mayor variabilidad y potencial para recompensas más altas debido a su naturaleza más exploratoria. Esto ilustra las distintas filosofías de aprendizaje de ambos algoritmos y cómo se manifiestan en el rendimiento de los robots en el entorno de entrenamiento multi-robot.

6.3.2.2. Entrenamiento 2

En la segunda simulación se hace uso de los parámetros que confieren el peor rendimiento al aprendizaje por refuerzo. Por ello, en el segundo entrenamiento realizado para entrenar a un sistema de dos robots utilizando el algoritmo Sarsa, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,9	0,1	0,9	0,999

Tabla 6-14. Parámetros del algoritmo Sarsa utilizados en el segundo entrenamiento multi-robot

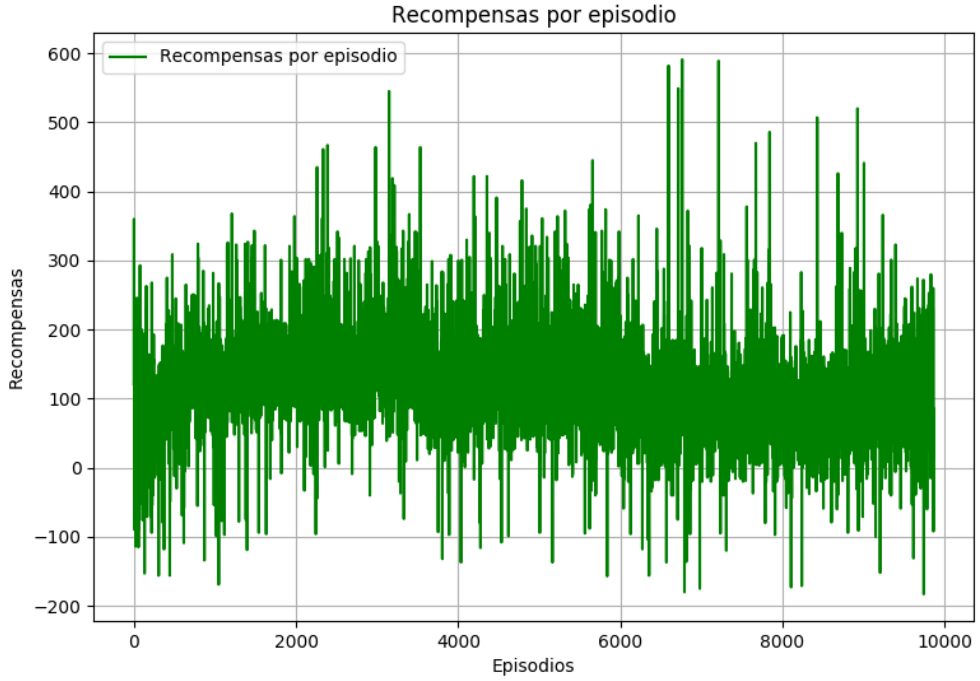


Figura 6-73. Recompensas por episodio en el segundo entrenamiento multi-robot con Sarsa

La figura 6-73, que recoge la gráfica de recompensas por episodio utilizando los parámetros alfa 0,9 y gamma 0,1, muestra una tendencia general en la que las recompensas alcanzan valores altos en ciertos episodios, llegando incluso a 600. Sin embargo, esta tendencia no es ascendente a lo largo del tiempo. Los robots parecen encontrar estrategias que les permiten obtener recompensas significativas rápidamente debido al elevado valor de alfa, que fomenta un aprendizaje rápido. Sin embargo, el bajo valor de gamma, que reduce la importancia de las recompensas futuras, resulta en una falta de mejora sostenida en las recompensas a largo plazo. Esto indica que, aunque los robots son capaces de explotar recompensas inmediatas eficientemente, no están desarrollando estrategias que mantengan un rendimiento elevado de manera constante.

En contraste, la gráfica de recompensas por episodio para Sarsa con alfa 0,1 y gamma 0,8 mostraba una tendencia en la que las recompensas se mantenían dentro de un rango más estable y consistentemente positivo. Las recompensas estaban más concentradas en valores positivos y mostraban una tendencia más sostenible.

La comparación entre ambos conjuntos de parámetros revela que, con Sarsa usando alfa 0,9 y gamma 0,1, los robots experimentan episodios con recompensas relativamente altas, pero sin una tendencia ascendente clara y sostenida en el tiempo, debido a su enfoque en la rápida explotación de recompensas inmediatas. Por otro lado, con Sarsa utilizando alfa 0,1 y gamma 0,8, las recompensas mostraban una tendencia más estable y consistentemente positiva, resultando en un rendimiento más sostenido. Esto se debe a un equilibrio más adecuado entre la velocidad de aprendizaje y la consideración de recompensas futuras, lo que permite a los robots desarrollar estrategias más robustas y sostenibles. En conclusión, mientras que un alfa alto y gamma bajo pueden proporcionar beneficios rápidos, un alfa bajo y gamma alto favorecen un rendimiento más estable y confiable a largo plazo.

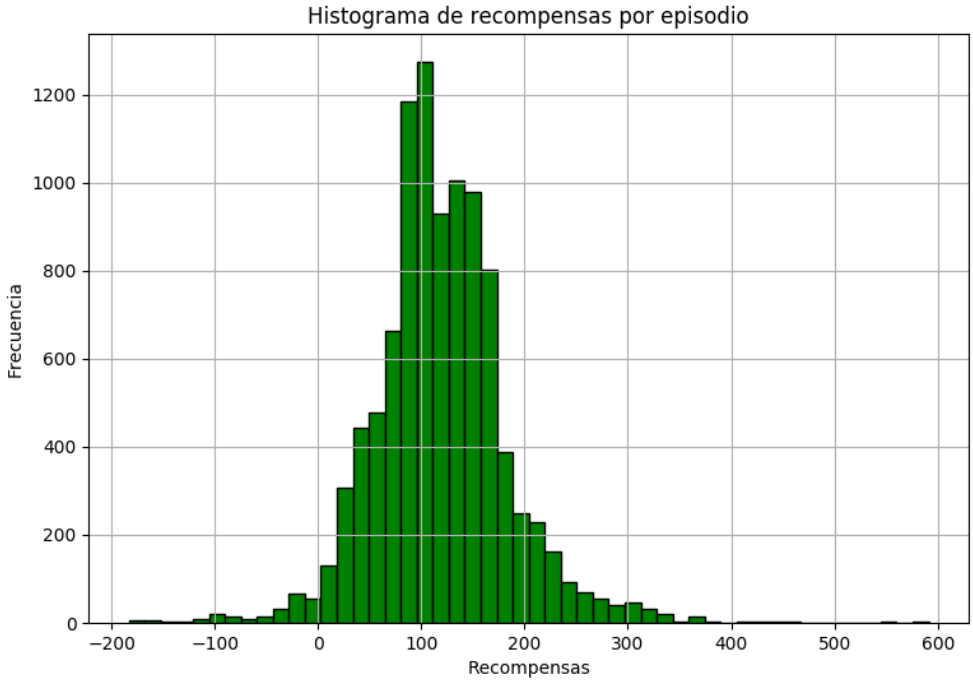


Figura 6-74. Histograma de recompensas por episodio en el segundo entrenamiento multi-robot con Sarsa

Para el entrenamiento multi-robot con Sarsa utilizando los parámetros alfa 0,9 y gamma 0,1, el histograma de recompensas por episodio, que se presenta en la figura 6-74, muestra una distribución amplia y dispersa. La mayoría de las recompensas están concentradas entre 0 y 200, pero también se observa una cantidad significativa de episodios con recompensas negativas y algunos episodios que superan los 400 en recompensas. Esta amplia distribución indica una alta variabilidad en las recompensas, lo cual refleja que los robots están adaptándose rápidamente a recompensas inmediatas gracias al alto valor de alfa. Sin embargo, la presencia de muchos episodios con recompensas bajas o negativas sugiere que las estrategias aprendidas no son sostenibles a largo plazo.

En contraste, el histograma correspondiente a Sarsa con alfa 0,1 y gamma 0,8 presentaba una distribución más concentrada alrededor de valores positivos. La mayoría de las recompensas se encontraban en el rango de 0 a 200, con un notable pico en el intervalo de 100 a 150. Sin embargo, también se observaba un pequeño pico en recompensas de valor 300, lo que indicaba un mayor número de recompensas ligeramente elevadas. El bajo alfa indicaba un aprendizaje más lento, pero más estable, mientras que el alto gamma aseguraba que las recompensas futuras fueran tenidas en cuenta en las decisiones actuales. Esto conducía a un rendimiento más equilibrado y sostenible, con estrategias que maximizaban las recompensas a mediano y largo plazo.

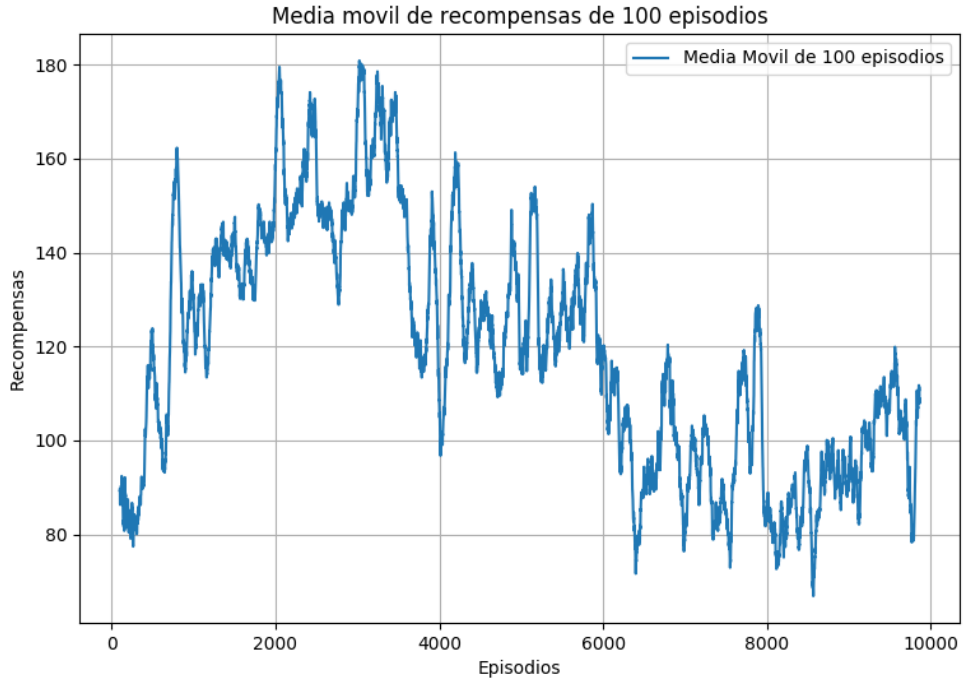


Figura 6-75. Medias móviles de las recompensas en el segundo entrenamiento multi-robot con Sarsa

La figura 6-75 muestra una tendencia ascendente inicial que alcanza picos cercanos a 180. Sin embargo, a medida que avanza el entrenamiento, la tendencia se estabiliza con grandes fluctuaciones, que reflejan una falta de consistencia en las estrategias aprendidas por los robots. Estas fluctuaciones indican que, aunque los robots pueden encontrar estrategias efectivas rápidamente debido al alto valor de alfa, la falta de consideración de recompensas futuras, representada por el bajo gamma, impide que estas estrategias sean sostenibles a largo plazo. Como resultado, el rendimiento de los robots muestra mejoras rápidas, pero no consistentes, lo que afecta negativamente a la estabilidad del sistema.

En contraste, en entrenamiento de Sarsa con alfa 0,1 y gamma 0,8 también mostraba una tendencia ascendente, pero de manera más gradual y sostenida, alcanzando picos de hasta 200. La media móvil revelaba menos fluctuaciones abruptas, lo que indicaba que los robots estaban desarrollando estrategias más estables y sostenibles a lo largo del tiempo.

Es fácil comprobar igualmente que, al comparar los resultados obtenidos en este caso con los obtenidos en el caso de un único robot en estas mismas condiciones, las recompensas obtenidas en este caso son mayores, así como el rendimiento general del aprendizaje por refuerzo. Esto se debe a que, como se comentó previamente, el entorno multi-robot se beneficia de una alta tasa de aprendizaje y un bajo factor de descuento (alfa 0,9 y gamma 0,1), mientras que el entorno de un solo robot se favorece con una baja tasa de aprendizaje y un alto enfoque en el futuro (alfa 0,1 y gamma 0,8). Estas diferencias resaltan la importancia de ajustar los parámetros de aprendizaje según el entorno y las interacciones.

6.3.2.3. Entrenamiento 3

En la tercera simulación multi-robot empleando el algoritmo Sarsa se utilizará de nuevo la combinación más favorable en el entrenamiento. Por ello, se han utilizado los siguientes parámetros:

Alfa	Gamma	Épsilon	Descuento de épsilon
0,1	0,8	0,9	0,999

Tabla 6-15. Parámetros del algoritmo Sarsa utilizados en el tercer entrenamiento multi-robot

En este caso, el entrenamiento se realizará en un entorno más amplio para analizar cómo la restricción de espacio afecta al rendimiento de los robots, de igual manera como se hizo anteriormente. En este caso, el obstáculo central simplemente se trata de un cuadrado que los robots deben bordear para llegar al punto de partida y completar el laberinto.

Las trayectorias seguidas por los dos robots en cada uno de los episodios se muestran a continuación, con el objetivo de poder visualizar el comportamiento de los agentes antes de pasar a analizar las recompensas obtenidas.

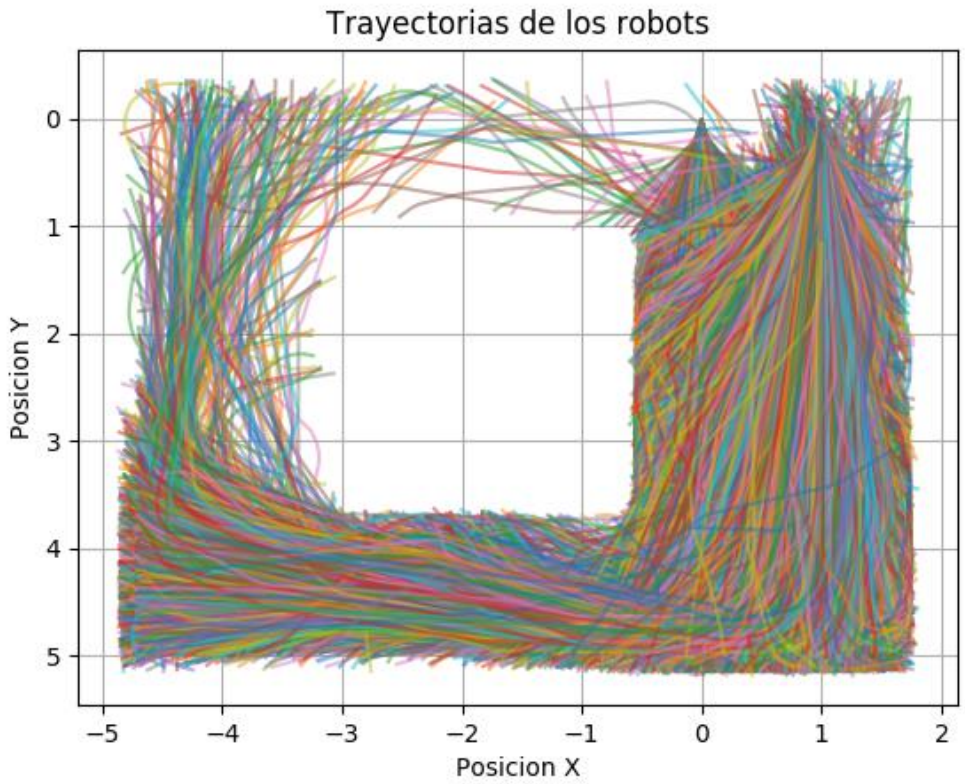


Figura 6-76. Trayectorias de los robots en cada uno de los episodios del entrenamiento múltiple ancho con Sarsa

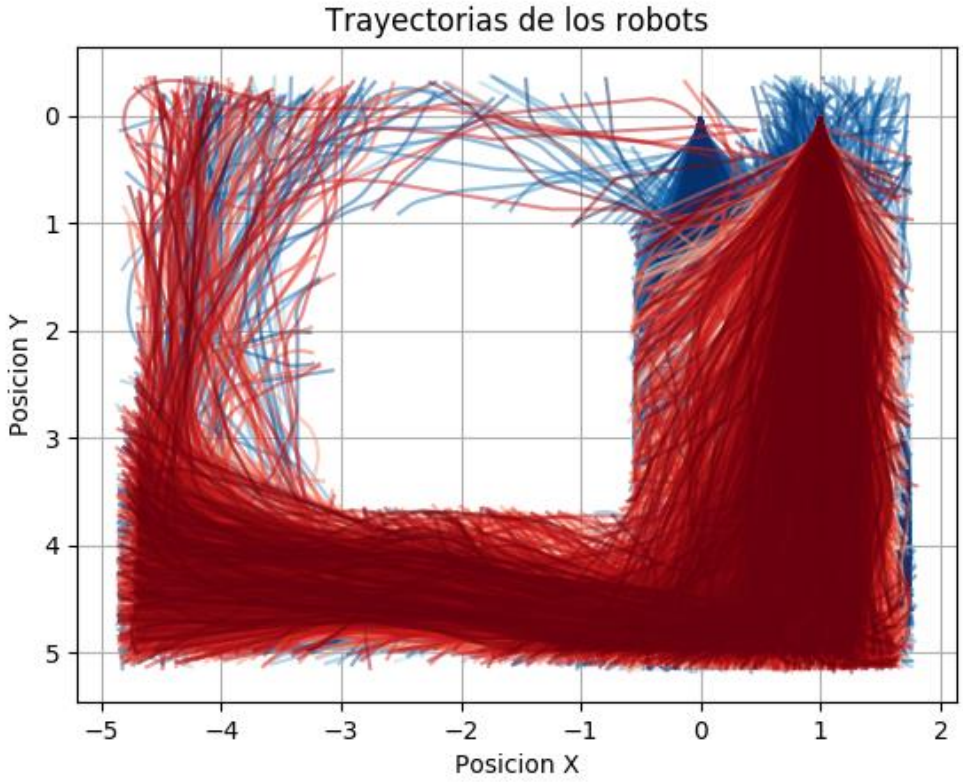


Figura 6-77. Trayectorias de los robots en cada episodio del entrenamiento múltiple ancho con Sarsa por colores

En las figuras 6-76 y 6-77 se muestran las trayectorias llevadas a cabo por cada uno de los robots a lo largo del entrenamiento. En contraste con el escenario anterior, donde el entorno era un laberinto con espacio limitado, en este caso se observa que en varios episodios los robots logran llegar a la meta y completar el recorrido.

Además, si se compara este entrenamiento con el realizado en el mismo entorno utilizando Q-Learning, se puede observar que la política aplicada por el algoritmo Sarsa permite que los robots completen un mayor número de episodios con avances más significativos en el recorrido. No obstante, también hay un gran número de episodios que se desvía y vuelve a la posición original sin recorrer el camino deseado.

Esto se debe a que Sarsa es un algoritmo basado en la política on-policy, lo que significa que actualiza sus valores de acción-estado basándose en la política que realmente está siguiendo el agente durante el entrenamiento. Como resultado, tiende a ser más conservador y ajustarse a las acciones que ha experimentado, lo que puede facilitar avances más consistentes en ciertos episodios. Sin embargo, este enfoque también puede llevar a que el robot siga explorando acciones subóptimas, lo que explica los episodios en los que se desvía o regresa a su posición inicial sin completar el recorrido deseado. Por otro lado, Q-Learning, al ser un algoritmo off-policy, tiende a explorar más agresivamente las acciones óptimas en cada estado, lo que puede reducir estos comportamientos erráticos, pero a costa de una menor estabilidad en el corto plazo.

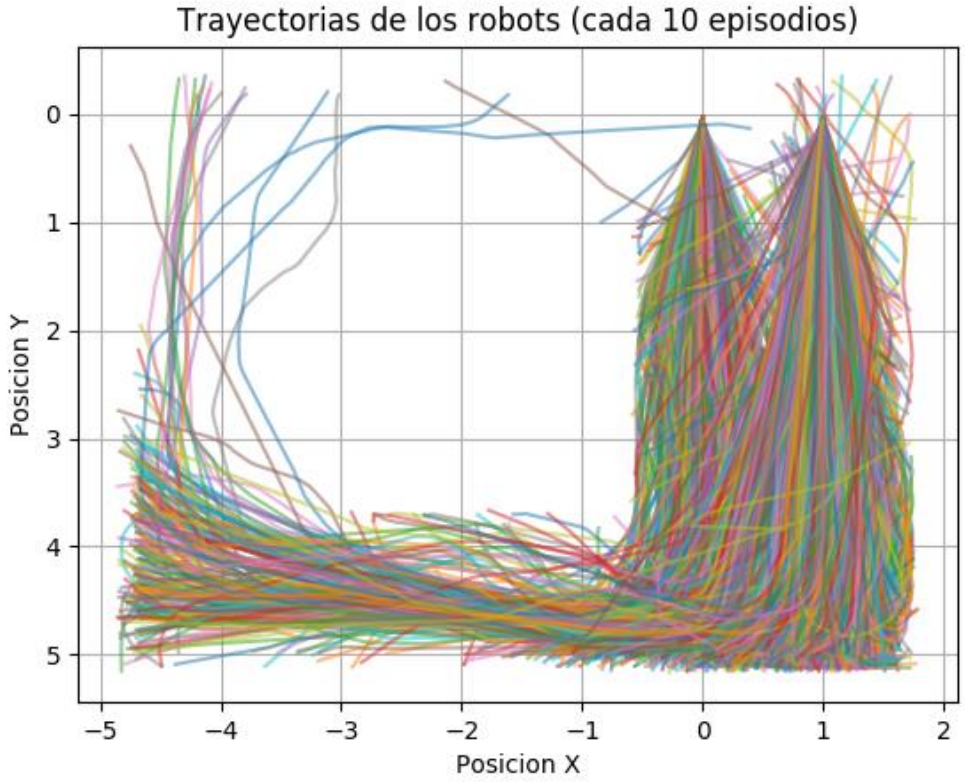


Figura 6-78. Trayectorias de los robots en uno de cada cien episodios del entrenamiento múltiple ancho con Sarsa

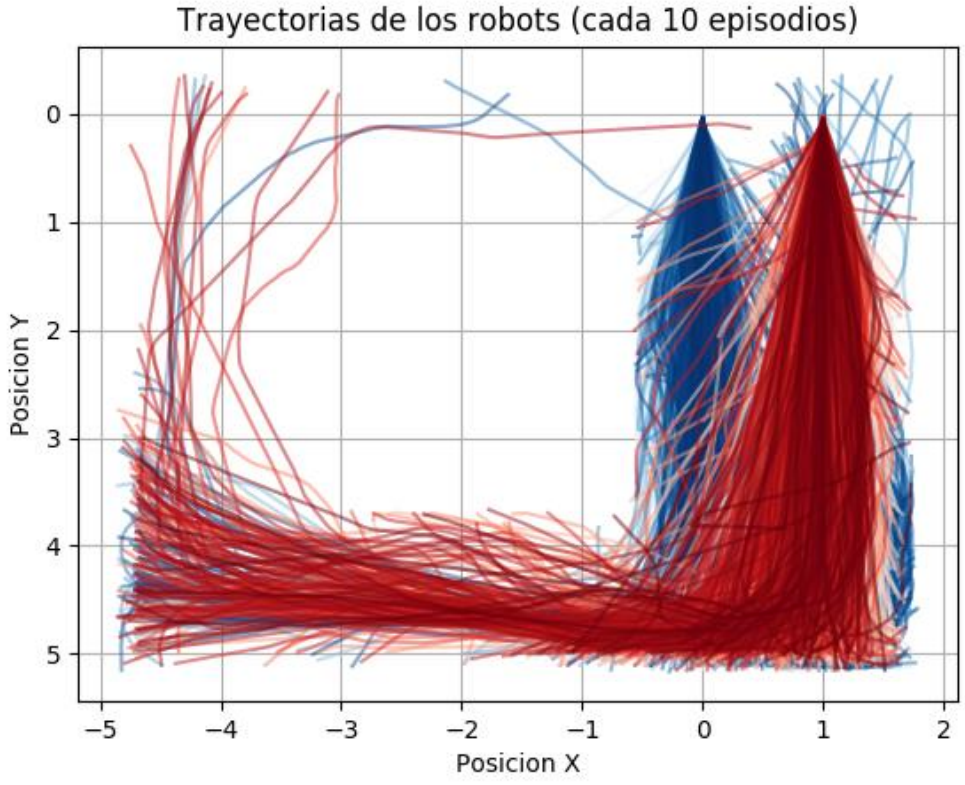


Figura 6-79. Trayectorias de los robots en uno de cada cien episodios del entrenamiento múltiple ancho con Sarsa por colores

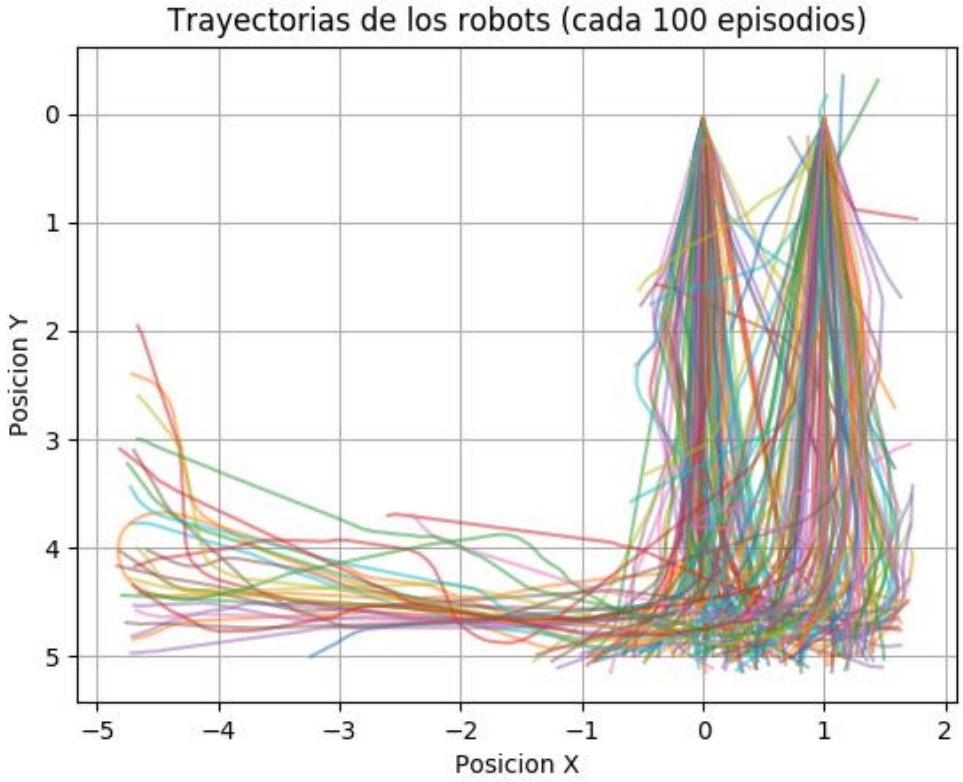


Figura 6-80. Trayectorias de los robots en uno de cada mil episodios del entrenamiento múltiple ancho con Sarsa

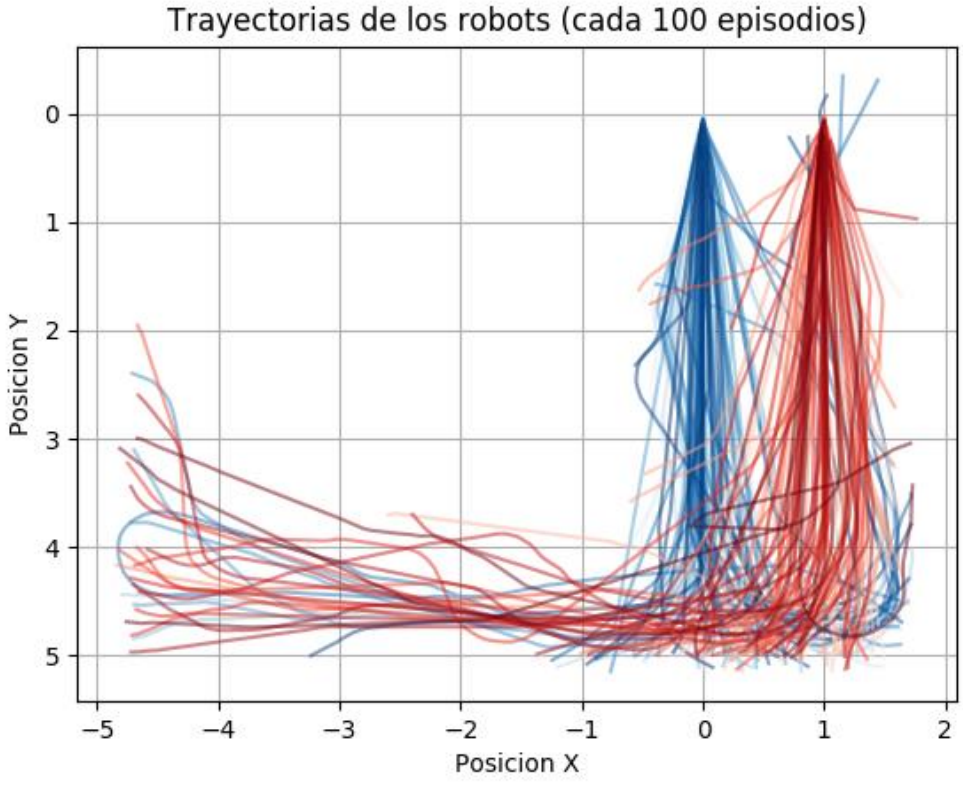


Figura 6-81. Trayectorias de los robots en uno de cada mil episodios del entrenamiento múltiple ancho con

Sarsa por colores

Se han incluido estas cuatro figuras adicionales para ofrecer una perspectiva más completa sobre el progreso del aprendizaje del robot a través del tiempo.

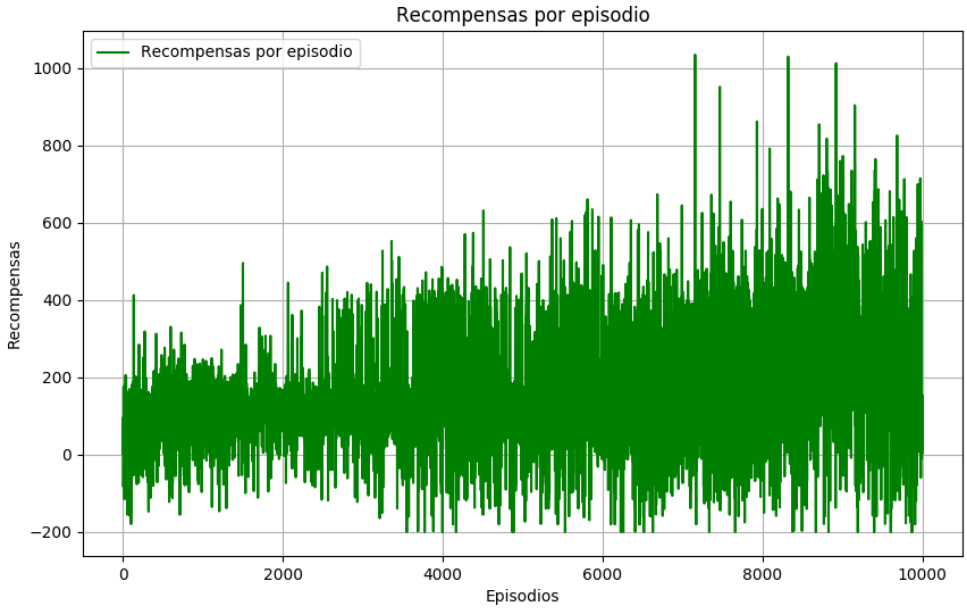


Figura 6-82. Recompensas por episodio en el tercer entrenamiento multi-robot con Sarsa

El análisis de los resultados del entrenamiento con Sarsa revela un comportamiento bastante similar en comparación con el obtenido mediante Q-Learning. En general, las recompensas por episodio en el entrenamiento con Sarsa presentan un rango que va aumentando gradualmente, con valores que tienden a situarse en su mayoría entre 100 y 400. Esta tendencia refleja una mejora constante en el rendimiento, pero con menos episodios en los que se logran recompensas extremas, a diferencia de Q-Learning, donde se observaron picos significativos y una mayor variabilidad.

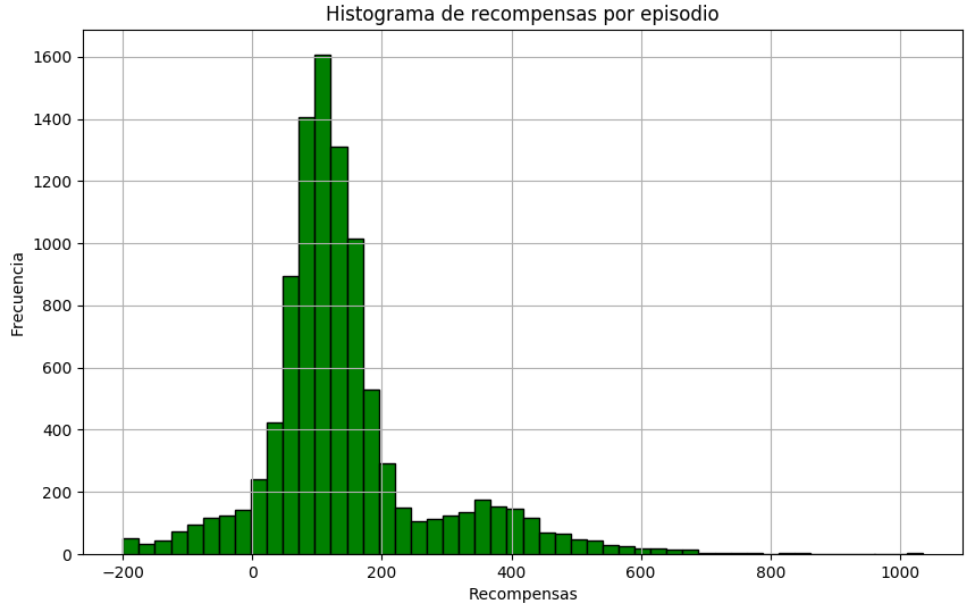


Figura 6-83. Histograma de recompensas por episodio en el tercer entrenamiento multi-robot con Sarsa

El histograma de recompensas en Sarsa es similar al observado en Q-Learning, lo que indica que ambos

algoritmos encuentran desafíos y oportunidades en el entorno multi-robot. Sin embargo, los picos principales se ubican alrededor de 100 y 400, con frecuencias de 1600 y 200 episodios respectivamente, lo que sugiere una distribución más concentrada de los resultados en comparación con Q-Learning, que mostró una mayor dispersión con picos más amplios en torno a 150 y 400. Esto indica que Sarsa genera un desempeño más uniforme y menos fluctuante, lo que podría deberse a su enfoque on-policy, que fomenta una exploración más conservadora y estable.

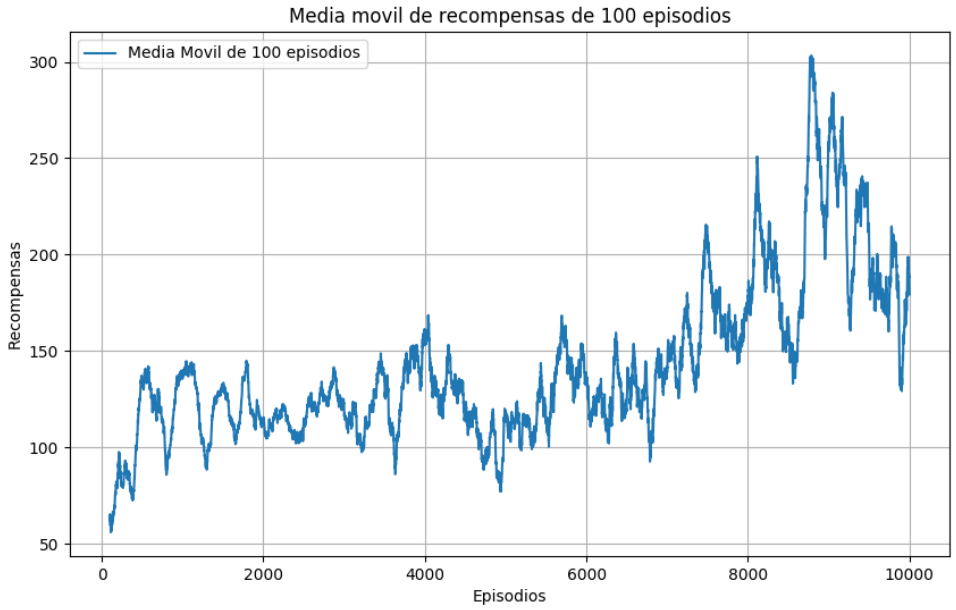


Figura 6-84. Medias móviles de las recompensas en el tercer entrenamiento multi-robot con Sarsa

En cuanto a las medias móviles de las recompensas, el entrenamiento con Sarsa muestra una fase de estancamiento entre los episodios 0 y 6000, donde las recompensas se mantienen entre 100 y 150, lo que contrasta con Q-Learning, que tiene una mejora progresiva más temprana. Sin embargo, a partir del episodio 6000, Sarsa muestra un incremento sostenido hasta alcanzar una media móvil de 300, lo que indica que, si bien el proceso de aprendizaje es más lento al principio, una vez que el algoritmo comienza a mejorar, lo hace de manera consistente. En contraste, Q-Learning, aunque muestra una tendencia ascendente más marcada desde el inicio, presenta una mayor variabilidad a lo largo del entrenamiento, lo que puede resultar en episodios de rendimiento excepcional, pero también en caídas pronunciadas.

7 CONCLUSIONES

En este punto final del estudio, se presenta un análisis comparativo exhaustivo de dos algoritmos de aprendizaje por refuerzo, evaluando no solo su rendimiento en diferentes escenarios, sino también la influencia de la variación de sus parámetros internos.

Este análisis se extiende a diferentes entornos de aplicación, abarcando tanto un entorno estático, donde un único robot opera de manera independiente, como un entorno dinámico, caracterizado por la interacción entre múltiples robots.

El objetivo es identificar las configuraciones óptimas y entender cómo las dinámicas del entorno y la parametrización afectan el desempeño de los algoritmos, proporcionando así una guía integral para la implementación efectiva de soluciones de aprendizaje en robótica.

7.1. Comparación parámetros

La elección de los parámetros en los algoritmos de aprendizaje por refuerzo, en particular alfa (la tasa de aprendizaje), gamma (el factor de descuento) y épsilon (la tasa de exploración-explotación), tiene un impacto profundo en el rendimiento y la estabilidad del aprendizaje. Los entrenamientos realizados han mostrado cómo diferentes configuraciones de estos parámetros afectan significativamente la capacidad del robot para aprender y optimizar su comportamiento.

7.1.1. Alfa bajo y gamma alto

Una configuración con un alfa bajo (0,1) y un gamma alto (0,8) ha demostrado promover un aprendizaje más estable y consistente. Este enfoque favorece la consideración de recompensas a largo plazo y permite que el robot ajuste sus estrategias de manera gradual, lo que se traduce en un rendimiento sostenido y predecible a lo largo del tiempo.

La tasa de aprendizaje baja asegura que los valores de Q se actualicen de manera gradual, proporcionando estabilidad y evitando fluctuaciones bruscas en el aprendizaje. Este tipo de configuración es particularmente efectiva para lograr un equilibrio entre la exploración y la explotación, ya que el robot no exagera nuevas experiencias y, en cambio, consolida de manera más estable sus aprendizajes anteriores.

Además, el gamma alto incentiva la maximización de las recompensas futuras, promoviendo estrategias que son beneficiosas a largo plazo en lugar de centrarse solo en recompensas inmediatas. Esta combinación ha mostrado en los entrenamientos una tendencia ascendente clara y sostenida en las recompensas, reflejando un comportamiento más controlado y menos volátil.

7.1.2. Alfa alto y gamma bajo

Por otro lado, una configuración con un alfa alto (0,9) y un gamma bajo (0,1) facilita un aprendizaje mucho más rápido, pero con una mayor inestabilidad. Este enfoque se centra en recompensas inmediatas, lo que puede llevar

a una rápida explotación de estrategias efectivas a corto plazo. Sin embargo, la falta de consideración de las recompensas futuras a menudo resulta en una falta de sostenibilidad y en un desempeño errático a largo plazo. Los entrenamientos han mostrado que las estrategias aprendidas bajo esta configuración tienden a ser más volátiles y menos confiables, lo que se refleja en una mayor variabilidad en las recompensas.

La alta tasa de aprendizaje provoca que los valores de Q se actualicen rápidamente con cada nueva experiencia, lo que puede llevar a un comportamiento inestable y errático, ya que el robot se adapta demasiado rápido a las nuevas experiencias sin dar suficiente consideración a la información acumulada.

Además, un gamma bajo significa que las recompensas futuras tienen poca influencia en las decisiones actuales, lo que lleva a una preferencia por las recompensas inmediatas y a estrategias que no son sostenibles a largo plazo.

7.1.3. Variación de épsilon

La variación de épsilon también ha jugado un papel muy importante en los resultados observados durante los entrenamientos. Épsilon controla el equilibrio entre exploración y explotación, es decir, la probabilidad de que el robot elija una acción aleatoria en lugar de la mejor acción conocida.

En los entrenamientos con una épsilon alto inicial y un descuento gradual (por ejemplo, una épsilon inicial de 0,9 con un descuento de 0,999), el robot comienza explorando intensamente el espacio de posibles acciones, lo que le permite descubrir estrategias potencialmente valiosas. Sin embargo, conforme épsilon disminuye, el robot transita hacia la explotación de las estrategias más exitosas. Esta estrategia ha mostrado ser efectiva en configuraciones con alfa bajo y gamma alto, donde la exploración inicial ayuda a consolidar un aprendizaje robusto y estable.

Por el contrario, en configuraciones con alfa alto y gamma bajo, una disminución rápida de épsilon puede llevar a un enfoque demasiado temprano en la explotación, lo que, combinado con un aprendizaje rápido y un bajo énfasis en las recompensas futuras, resulta en estrategias que parecen exitosas a corto plazo pero que fallan en mantener un rendimiento sostenido.

Reducir el valor de épsilon a valores más bajos, como 0,9, 0,8 o 0,7, ha disminuye la cantidad de exploración en el proceso de aprendizaje del robot, enfocando sus acciones más en la explotación de las estrategias ya conocidas. Esta menor exploración ha limitado la capacidad del robot para descubrir nuevas estrategias potencialmente más efectivas, lo que ha resultado ser perjudicial. Cuando la exploración inicial es insuficiente, el robot consolida prematuramente una estrategia subóptima, afectando negativamente su adaptabilidad y capacidad para encontrar soluciones óptimas. Además, una épsilon baja acelera la transición hacia la explotación de recompensas inmediatas, especialmente en configuraciones con valores altos de alfa y bajos de gamma, lo que ha llevado a un rendimiento efectivo a corto plazo, pero insostenible a largo plazo.

7.1.4. Tablas comparativas

A continuación, se adjuntan tres tablas comparativas donde se pueden apreciar las diferencias en el comportamiento de los agentes a través de las medias móviles de las recompensas obtenidas.

Se presentan la variación en los parámetros alfa y gamma en la primera tabla, y el parámetro épsilon en la segunda tabla, ambas empleando el algoritmo Q-Learning. La tercera tabla recoge la variación de los parámetros alfa y gamma utilizando el algoritmo Sarsa. La simbología que aparece al lado del valor del parámetro indica si ese parámetro se incrementa, decrece o se mantiene igual con respecto al primer entrenamiento de la tabla, que se toma como referencia.

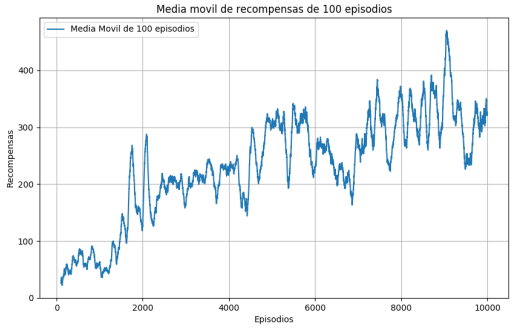
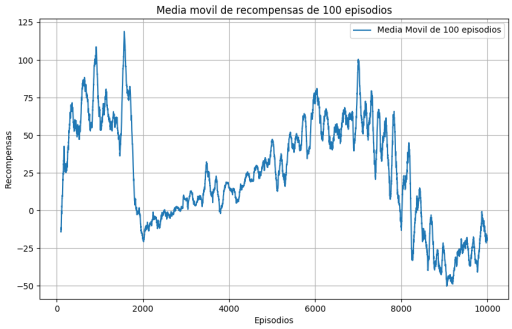
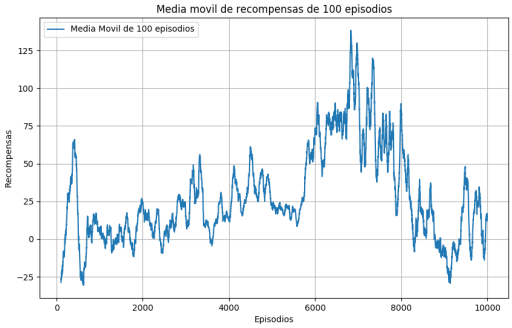
Q-Learning	Alfa	Gamma	Épsilon	Descuento de épsilon	Medias móviles
Simulación 1	0,1	0,8	0,9	0,999	
Simulación 2	0,9 ↑	0,1 ↓	0,9 =	0,999 =	
Simulación 3	0,5 ↑	0,5 ↓	0,9 =	0,999 =	

Tabla 7-1. Comparación de parámetros alfa y gamma en los entrenamientos con Q-Learning

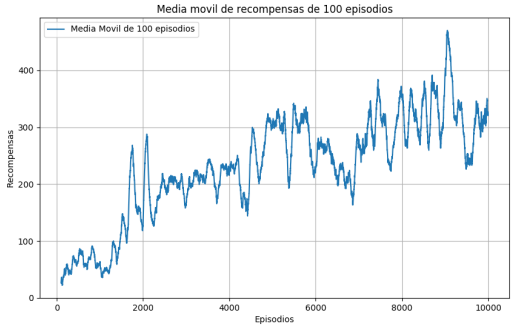
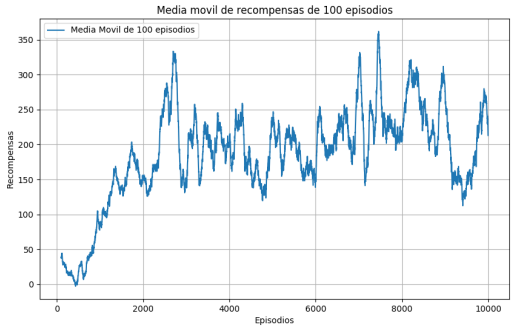
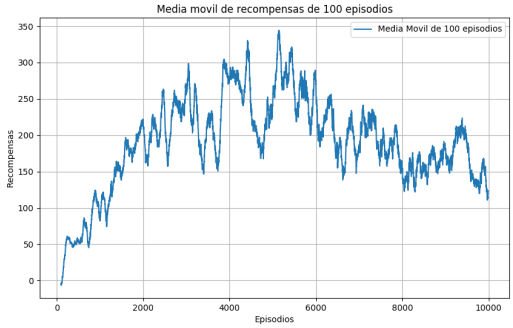
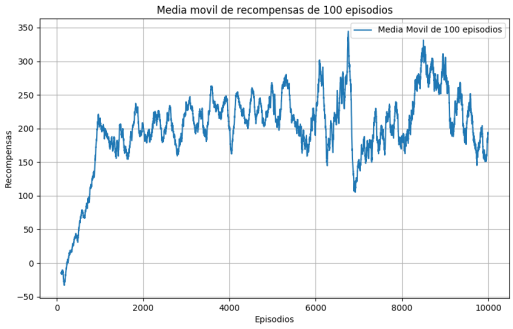
Q-Learning	Alfa	Gamma	Épsilon	Descuento de épsilon	Medias móviles
Simulación 1	0,1	0,8	0,9	0,999	
Simulación 4	0,1 =	0,8 =	0,9 =	0,9 ↓	
Simulación 5	0,1 =	0,8 =	0,9 =	0,8 ↓	
Simulación 6	0,1 =	0,8 =	0,9 =	0,7 ↓	

Tabla 7-2. Comparación de parámetro descuento de épsilon en los entrenamientos con Q-Learning

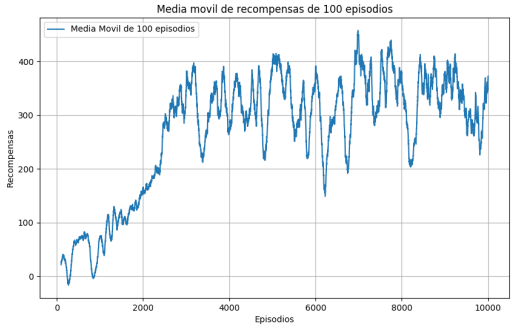
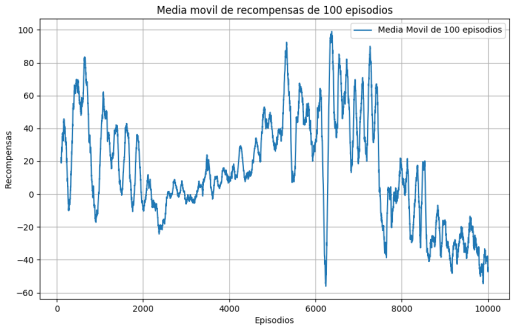
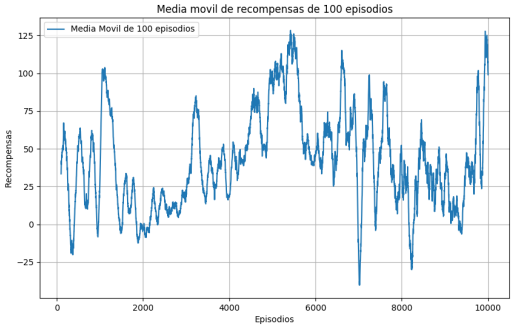
Sarsa	Alfa	Gamma	Épsilon	Descuento de épsilon	Medias móviles
Simulación 1	0,1	0,8	0,9	0,999	
Simulación 2	0,9 ↑	0,1 ↓	0,9 =	0,999 =	
Simulación 3	0,5 ↑	0,5 ↓	0,9 =	0,999 =	

Tabla 7-3. Comparación de parámetros alfa y gamma en los entrenamientos con Sarsa

7.2. Comparación algoritmos

Los diferentes algoritmos de aprendizaje por refuerzo, como Q-Learning y Sarsa, ofrecen enfoques distintos para el entrenamiento de robots, lo que se refleja en sus características y comportamientos particulares, así como en su impacto en el rendimiento general.

Q-Learning, conocido por su enfoque optimista y explorador, actualiza los valores de Q basándose en las mejores acciones posibles, es decir, aquellas que prometen las mayores recompensas futuras. Esta característica hace que Q-Learning sea particularmente efectivo en situaciones donde es importante explorar agresivamente el espacio de posibles acciones, buscando estrategias que maximicen las recompensas.

Sin embargo, esta misma agresividad en la exploración puede resultar en una mayor variabilidad en el rendimiento. A lo largo del entrenamiento, es común observar picos altos de recompensas en ciertos episodios, reflejando momentos en los que el robot descubre acciones extremadamente beneficiosas. No obstante, esta capacidad de alcanzar recompensas extremas a menudo viene acompañada de una falta de consistencia y

predictibilidad en el desempeño global, haciendo que Q-Learning sea más adecuado en escenarios donde la posibilidad de obtener recompensas muy altas es prioritaria, incluso si esto implica aceptar una mayor variabilidad en los resultados.

En comparación, Sarsa, que actualiza los valores de Q basándose en las acciones que el robot realmente toma, adopta un enfoque más conservador y estable. Este algoritmo prioriza la estabilidad en el proceso de aprendizaje, lo que se traduce en una menor variabilidad en las recompensas obtenidas a lo largo del tiempo. A diferencia de Q-Learning, que tiende a experimentar con diferentes acciones en busca de aquellas que maximicen las recompensas futuras, Sarsa se enfoca en consolidar las estrategias que el robot ya está utilizando, ajustándolas de manera gradual y segura.

Como resultado, la tendencia ascendente en las recompensas con Sarsa es generalmente más sostenida, lo que refleja una mayor consistencia en el aprendizaje y en el rendimiento del robot. Esta estabilidad hace que Sarsa sea especialmente adecuado para entornos donde la predictibilidad y la consistencia en el rendimiento son críticas, como en aplicaciones donde la fiabilidad del robot es más importante que la posibilidad de alcanzar recompensas esporádicamente altas.

En términos de influencia en el rendimiento, la naturaleza optimista de Q-Learning impulsa una exploración más agresiva de acciones potencialmente valiosas, lo que puede llevar a la identificación de estrategias altamente recompensadas, pero a costa de una mayor variabilidad y menos estabilidad. Por otro lado, la actualización basada en acciones reales en Sarsa promueve un aprendizaje más seguro y menos volátil, resultando en un desempeño más predecible y confiable a lo largo del tiempo. Esta diferencia fundamental entre ambos algoritmos destaca cómo la elección del algoritmo de aprendizaje por refuerzo puede depender en gran medida de los objetivos específicos del entrenamiento, ya sea que se valore la posibilidad de obtener recompensas altas con cierta inestabilidad o la necesidad de un rendimiento consistente y predecible.

7.2.1. Tablas comparativas

A continuación, se muestran una serie de tablas que recogen la comparación entre ambos algoritmos en tres situaciones distintas de aprendizaje, cada una con una configuración diferente de parámetros.

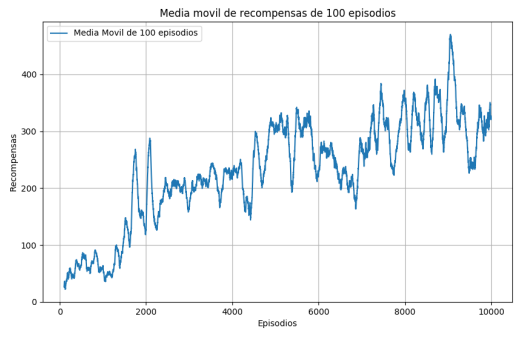
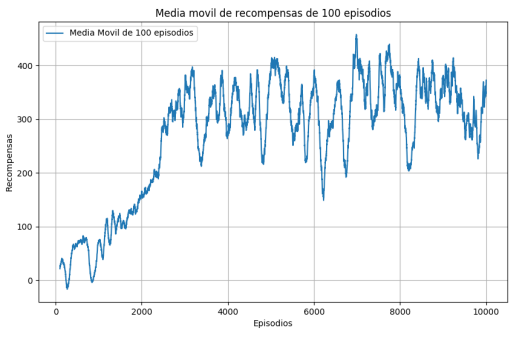
Alfa 0,1	**Gamma 0,8**	**Épsilon 0,9**	**Descuento de épsilon 0,999**
Q-Learning		Sarsa	
			

Tabla 7-4. Comparación de algoritmos con alfa 0,1 y gamma 0,8

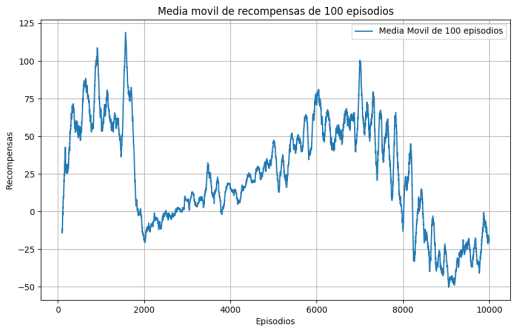
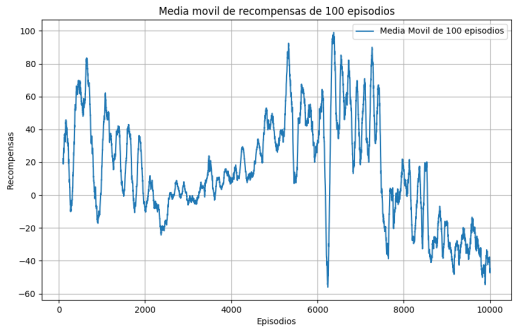
Alfa 0,9	Gamma 0,1	Épsilon 0,9	Descuento de épsilon 0,999
Q-Learning		Sarsa	
			

Tabla 7-5. Comparación de algoritmos con alfa 0,9 y gamma 0,1

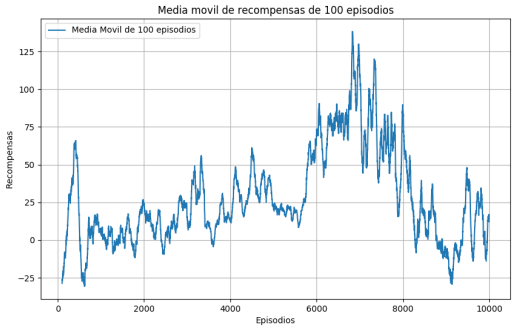
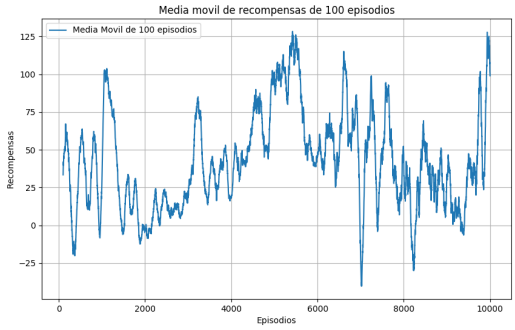
Alfa 0,5	Gamma 0,5	Épsilon 0,9	Descuento de épsilon 0,999
Q-Learning		Sarsa	
			

Tabla 7-6. Comparación de algoritmos con alfa 0,5 y gamma 0,5

7.3. Comparación entorno estático-dinámico / único robot-multi-robot

El entrenamiento en entornos con un solo robot y en entornos multi-robot presenta diferencias marcadas en términos de complejidad y resultados, con implicaciones significativas para el rendimiento y la estabilidad del aprendizaje.

En el caso de un solo robot, los resultados han mostrado una clara ventaja, con una menor variabilidad en las recompensas y un desempeño más consistente. En este entorno, el robot tiene la capacidad de enfocarse completamente en la optimización de sus propias estrategias, sin la interferencia de otros agentes. Esto permite que el aprendizaje sea más directo y controlado, lo que se refleja en una tendencia ascendente y sostenida en las recompensas obtenidas. La predictibilidad y estabilidad logradas en este tipo de entrenamiento demuestran que, sin la necesidad de competencia con otros robots, el proceso de aprendizaje puede centrarse en la maximización de recompensas de manera más efectiva y eficiente.

En contraste, el entrenamiento en un entorno multi-robot ha demostrado ser considerablemente más desafiante, con resultados de recompensas que han sido notablemente peores en comparación con el entorno de un solo

robot. La necesidad de cooperar entre múltiples robots introduce una mayor complejidad, lo que dificulta la estabilidad del rendimiento. Los robots en un entorno multi-robot no solo deben optimizar sus propias estrategias, sino que también deben gestionar las interacciones con otros robots, lo que a menudo lleva a resultados impredecibles y fluctuantes. Aunque existe un potencial para alcanzar recompensas altas en episodios específicos, este potencial rara vez se realiza de manera consistente.

En lugar de la tendencia ascendente clara observada en el entrenamiento de un solo robot, los resultados en entornos multi-robot muestran una mayor variabilidad y, en muchos casos, una falta de mejora sostenida a lo largo del tiempo. La interacción constante y la competencia por recursos entre robots contribuyen a este rendimiento inferior, subrayando los desafíos adicionales y la mayor complejidad asociada al entrenamiento en entornos multi-robot.

No obstante, se ha observado que los parámetros de aprendizaje influyen significativamente en estos resultados. En un entorno multi-robot, una alta tasa de aprendizaje y un bajo factor de descuento permiten adaptaciones rápidas a las recompensas inmediatas. Esto resulta en un rendimiento superior en comparación con un entorno de un solo robot bajo las mismas condiciones. Sin embargo, con una baja tasa de aprendizaje y un alto enfoque en el futuro, el entorno multi-robot sufre debido a la complejidad, mientras que el entorno de un solo robot puede seguir optimizando de manera efectiva.

Estas observaciones subrayan la importancia de ajustar los parámetros de aprendizaje según el entorno específico y las interacciones predominantes, destacando que los entornos multi-robot requieren una configuración diferente de parámetros para maximizar su rendimiento.

7.3.1. Tablas comparativas

A continuación, se muestran una serie de tablas que recogen la comparación entre entrenamientos en entorno estático y dinámico en tres situaciones distintas de aprendizaje, cada una con una configuración diferente de parámetros.

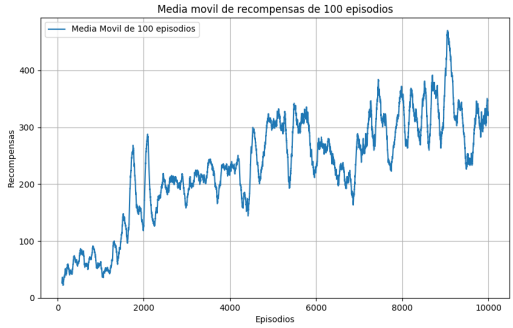
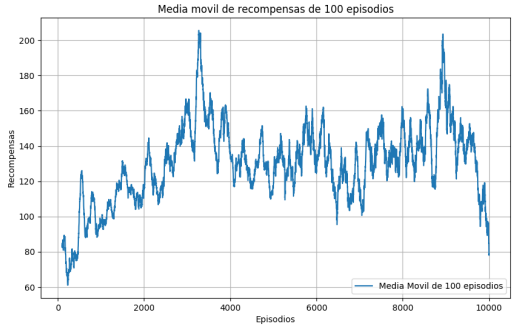
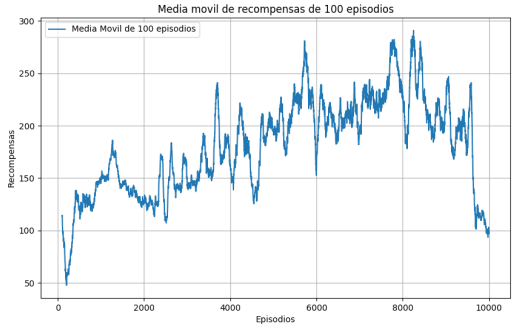
<table>
<tr><td>Q-Learning</td><td>Alfa 0,1</td><td colspan="2">Gamma 0,8</td><td>Épsilon 0,9</td><td>Descuento de épsilon 0,999</td></tr>
<tr><td colspan="3">Único robot / Entorno estático</td><td colspan="3">Multi-robot / Entorno dinámico estrecho</td></tr>
<tr><td colspan="3"> 
</td><td colspan="3"> 
</td></tr>
<tr><td colspan="3"></td><td colspan="3">Multi-robot / Entorno dinámico ancho</td></tr>
<tr><td colspan="3"></td><td colspan="3"> 
</td></tr>
</table>

Tabla 7-7. Comparación entorno estático y dinámico con Q-Learning, alfa 0,1 y gamma 0,8

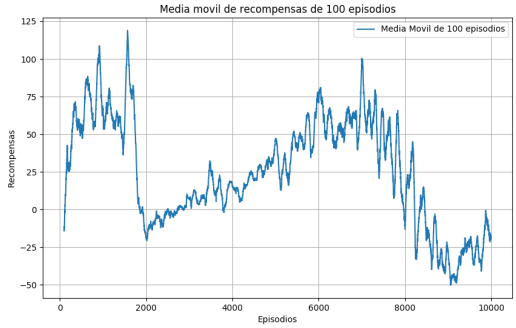
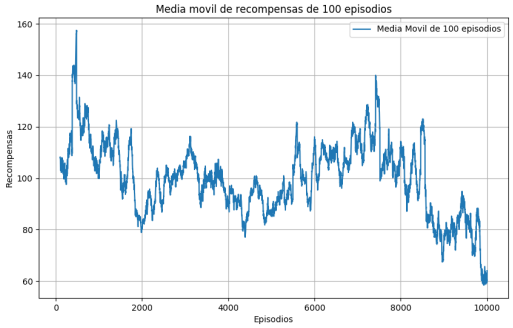
Q-Learning	**Alfa 0,9**	**Gamma 0,1**	**Épsilon 0,9**	**Descuento de épsilon 0,999**
Único robot / Entorno estático		Multi-robot / Entorno dinámico		
				

Tabla 7-8. Comparación entorno estático y dinámico con Q-Learning, alfa 0,9 y gamma 0,1

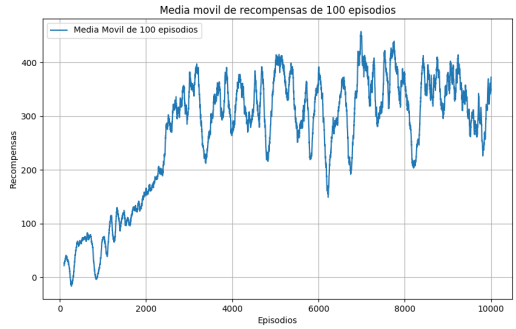
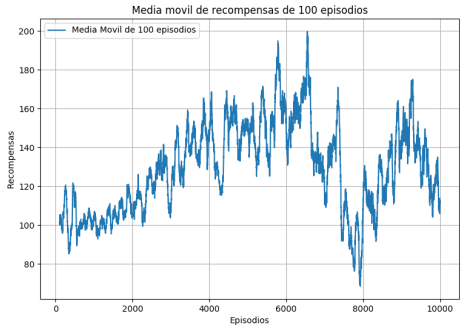
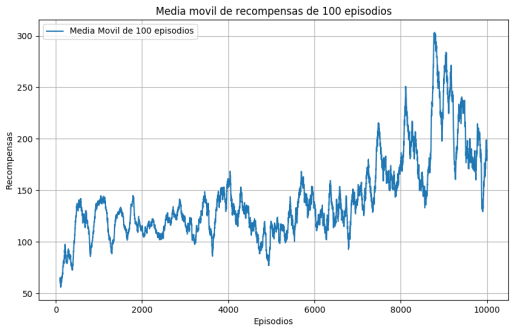
Sarsa	**Alfa 0,1**	**Gamma 0,8**	**Épsilon 0,9**	**Descuento de épsilon 0,999**
Único robot / Entorno estático		Multi-robot / Entorno dinámico estrecho		
				
		Multi-robot / Entorno dinámico ancho		
				

Tabla 7-9. Comparación entorno estático y dinámico con Sarsa, alfa 0,1 y gamma 0,8

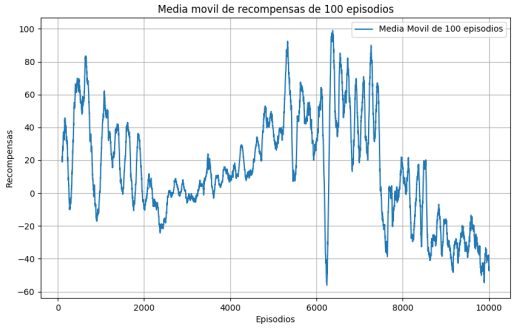
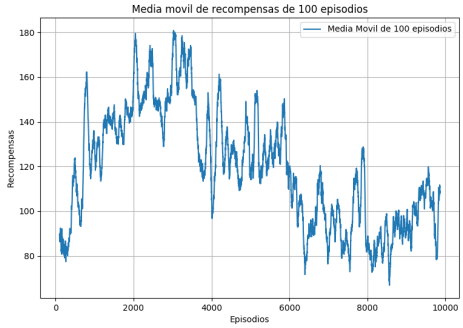
Sarsa	**Alfa 0,9**	**Gamma 0,1**	**Épsilon 0,9**	**Descuento de épsilon 0,999**
Único robot / Entorno estático		Multi-robot / Entorno dinámico		
				

Tabla 7-10. Comparación entorno estático y dinámico con Sarsa, alfa 0,9 y gamma 0,1

8 LÍNEAS DE TRABAJO FUTURAS

El presente trabajo ha explorado el entrenamiento de robots utilizando los algoritmos de aprendizaje por refuerzo Q-Learning y Sarsa, evaluando su rendimiento tanto en escenarios de entrenamiento individual como con obstáculos móviles.

Los resultados obtenidos han permitido identificar las fortalezas y debilidades de cada algoritmo en diferentes configuraciones de parámetros y entornos. Sin embargo, una importante área de investigación que no se ha abordado en este trabajo es la implementación del algoritmo Deep Q-Learning, una técnica avanzada que utiliza redes neuronales profundas para aproximar la función Q, lo que podría ofrecer mejoras significativas en el rendimiento y la escalabilidad del aprendizaje por refuerzo.

La implementación y evaluación de Deep Q-Learning representa una de las direcciones más prometedoras para futuras investigaciones, pero no es la única. Existen varias líneas de trabajo que pueden ampliar y profundizar en los hallazgos de este estudio, abordando desde la escalabilidad y complejidad de los entornos hasta la optimización de parámetros y el uso de técnicas avanzadas de aprendizaje. Estas líneas de trabajo no solo contribuirán a mejorar la eficacia de los algoritmos de aprendizaje por refuerzo, sino que también ofrecerán conocimientos valiosos sobre su aplicabilidad en escenarios más complejos y dinámicos.

Además, una línea de trabajo futura importante es la coordinación de múltiples robots mediante un sistema jerárquico en el que un robot líder realiza el aprendizaje y uno o varios robots seguidores ajustan sus acciones en función de las decisiones del líder. Este enfoque podría mejorar la eficiencia y la cohesión en tareas colaborativas, permitiendo una mayor adaptación y sincronización en entornos dinámicos y complejos.

8.1. Desarrollo de Deep Q-Learning

Este enfoque implica utilizar redes neuronales profundas para aproximar la función Q, permitiendo a los robots manejar estados de alta dimensionalidad y tomar decisiones más complejas. La implementación de Deep Q-Learning requiere la construcción de una red neuronal adecuada, la formulación del problema de aprendizaje por refuerzo en términos de entrada y salida para la red, y la adaptación del algoritmo de optimización para actualizar los pesos de la red basándose en las recompensas obtenidas.

8.1.1. Comparación de algoritmos

Una vez implementado Deep Q-Learning, es fundamental realizar comparaciones detalladas entre este algoritmo

y los algoritmos tradicionales de Q-Learning y Sarsa. Estas comparaciones deben centrarse en métricas clave como el rendimiento general en tareas específicas, la estabilidad del aprendizaje a lo largo del tiempo y la escalabilidad del algoritmo en entornos más complejos. Evaluar cómo este algoritmo se comporta en comparación con Q-Learning y Sarsa permitirá identificar sus ventajas y desventajas, proporcionando una comprensión más profunda de cuándo y cómo utilizar cada algoritmo para obtener los mejores resultados en el entrenamiento de robots.

8.1.2. Ajuste de parámetros

El ajuste de parámetros es una etapa esencial en la optimización del rendimiento de Deep Q-Learning. Diferentes configuraciones, como el tamaño de la red neuronal, la tasa de aprendizaje y el valor de gamma, pueden tener un impacto significativo en la eficacia del algoritmo. Explorar cómo estos parámetros afectan el rendimiento permitirá identificar las configuraciones óptimas para diversas tareas y entornos. Además, el ajuste de parámetros puede ayudar a mejorar la velocidad de convergencia y la estabilidad del aprendizaje, asegurando que el robot pueda adaptarse rápidamente a nuevos entornos y tareas sin sacrificar la precisión y la eficiencia.

8.1.3. Análisis de convergencia

Finalmente, el análisis de convergencia es esencial para entender cómo Deep Q-Learning se compara con Q-Learning y Sarsa en términos de velocidad y estabilidad de aprendizaje. Estudiar la convergencia implica evaluar la rapidez con la que cada algoritmo alcanza una política óptima y cómo se comporta el aprendizaje a medida que el entrenamiento progresa. Un análisis detallado de la convergencia permitirá identificar las fortalezas y debilidades de Deep Q-Learning, así como posibles áreas de mejora en su implementación. Comparar la estabilidad de este algoritmo con la de los algoritmos tradicionales también ayudará a comprender mejor su robustez y fiabilidad en diferentes escenarios de aprendizaje por refuerzo.

8.2. Coordinación de múltiples robots mediante un sistema de seguimiento

Una interesante extensión del estudio es la coordinación de múltiples robots mediante un sistema de aprendizaje jerárquico. En este enfoque, un robot líder se encarga de realizar el aprendizaje y tomar decisiones, mientras que uno o varios robots seguidores ajustan sus acciones en función de las decisiones del robot líder. Este sistema puede mejorar la eficiencia y la cohesión en tareas de coordinación, permitiendo que los robots seguidores se adapten dinámicamente a las estrategias aprendidas por el robot líder.

8.2.1. Implementación del sistema de aprendizaje jerárquico

La implementación de este sistema requiere desarrollar un mecanismo para que el robot líder pueda aprender y actualizar sus estrategias, mientras que los robots seguidores deben ser capaces de interpretar y seguir las acciones del líder. Esto incluye diseñar una arquitectura de comunicación efectiva y un protocolo para la sincronización y adaptación de los robots seguidores. Evaluar la efectividad del sistema en diferentes escenarios permitirá entender cómo la coordinación entre robots puede mejorar el rendimiento global.

8.2.2. Evaluación del rendimiento del sistema y optimización de la coordinación

Comparar el rendimiento del sistema de coordinación con el de sistemas no coordinados proporcionará una visión clara de las ventajas y desventajas del enfoque de aprendizaje jerárquico. Las métricas a considerar incluyen la eficiencia en la ejecución de tareas, la capacidad de adaptación a cambios en el entorno y la robustez del sistema frente a fallos en uno o varios robots. Además, el ajuste fino de parámetros es crucial para maximizar el rendimiento del sistema coordinado. Esto implica la calibración de los parámetros de aprendizaje del robot líder y la optimización de los mecanismos de seguimiento y adaptación de los robots seguidores. Explorar diferentes configuraciones y protocolos de coordinación permitirá identificar las mejores prácticas y ajustar el diseño del sistema para mejorar la eficacia en tareas colaborativas.

REFERENCIAS

[1] Ding, Z., Huang, Y., Yuan, H., & Dong, H. (2020). Introduction to Reinforcement Learning. En H. Dong, Z. Ding, & S. Zhang (Eds.), *Deep Reinforcement Learning: Fundamentals, Research and Applications* (pp. 47-123). Springer. https://doi.org/10.1007/978-981-15-4095-0_2

[2] AlMahamid, F., & Grolinger, K. (2021). Reinforcement Learning Algorithms: An Overview and Classification. *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 1-7. https://doi.org/10.1109/CCECE53047.2021.9569056

[3] Chadi, M.-A., & Mousannif, H. (2023). *Understanding Reinforcement Learning Algorithms: The Progress from Basic Q-learning to Proximal Policy Optimization* (arXiv:2304.00026). arXiv. https://doi.org/10.48550/arXiv.2304.00026

[4] Li, Y. (2018). *Deep Reinforcement Learning* (arXiv:1810.06339). arXiv. https://doi.org/10.48550/arXiv.1810.06339

[5] Uther, W. (2010). Markov Decision Processes. En C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 642-646). Springer US. https://doi.org/10.1007/978-0-387-30164-8_512

[6] Hu, M. (2023). Markov Decision Processes. En M. Hu (Ed.), *The Art of Reinforcement Learning: Fundamentals, Mathematics, and Implementations with Python* (pp. 21-46). Apress. https://doi.org/10.1007/978-1-4842-9606-6_2

[7] Ray, S., & Tadepalli, P. (2014). Model-Based Reinforcement Learning. En C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning and Data Mining* (pp. 1-6). Springer US. https://doi.org/10.1007/978-1-4899-7502-7_561-1

[8] Sutton, R. S., & Barto, A. G. (s. f.). *Reinforcement Learning: An Introduction*.

[9] Baxter, J., Tridgell, A., & Weaver, L. (1999). *TDLeaf(lambda): Combining Temporal Difference Learning with Game-Tree Search* (arXiv:cs/9901001). arXiv. https://doi.org/10.48550/arXiv.cs/9901001

[10] Sewak, M. (2019). Temporal Difference Learning, SARSA, and Q-Learning. En M. Sewak (Ed.), *Deep Reinforcement Learning: Frontiers of Artificial Intelligence* (pp. 51-63). Springer. https://doi.org/10.1007/978-981-13-8285-7_4

[11] Anas, H., Ong, W. H., & Malik, O. A. (2022). Comparison of Deep Q-Learning, Q-Learning and SARSA Reinforced Learning for Robot Local Navigation. En J. Kim, B. Englot, H.-W. Park, H.-L. Choi, H. Myung, J. Kim, & J.-H. Kim (Eds.), *Robot Intelligence Technology and Applications 6* (pp. 443-454). Springer International Publishing. https://doi.org/10.1007/978-3-030-97672-9_40

[12] *Q-learning—Wikipedia*. (s. f.). Recuperado 4 de julio de 2024, de https://en.wikipedia.org/wiki/Q-learning

[13] *DQN - AI Wiki—Artificial Intelligence, Machine Learning Wiki and Guide*. (s. f.). Recuperado 5 de julio de 2024, de https://aiwiki.ai/wiki/DQN

[14] *An Overview of Model-Based Reinforcement Learning | Shiyu Chen*. (s. f.). Recuperado 7 de julio de 2024, de https://www.chenshiyu.top/blog/2019/06/12/An-Overview-of-Model-Based-Reinforcement-Learning/

[15] *Gazebo—Docs: Sim Architecture*. (s. f.). Recuperado 8 de julio de 2024, de https://gazebosim.org/docs/harmonic/architecture

[16] *ROS Gazebo: Everything You Need To Know - Robotic Simulation Services*. (s. f.). Recuperado 8 de julio de 2024, de https://roboticsimulationservices.com/ros-gazebo-everything-you-need-to-know/

[17] *es/ROS/Conceptos—ROS Wiki*. (s. f.). Recuperado 8 de julio de 2024, de https://wiki.ros.org/es/ROS/Conceptos

[18] *Basic Usage—Gym Documentation*. (s. f.). Recuperado 8 de julio de 2024, de https://www.gymlibrary.dev/content/basic_usage/

[19] *Openai/gym*. (2024). [Python]. OpenAI. https://github.com/openai/gym (Obra original publicada en 2016)

[20] *Erlerobot/gym-gazebo*. (2024). [Python]. Erle Robotics. https://github.com/erlerobot/gym-gazebo (Obra original publicada en 2016)

[21] *Husarion/rosbot_description*. (2024). [Python]. Husarion. https://github.com/husarion/rosbot_description (Obra original publicada en 2022)

[22] *2019-tfm-ignacio-arranz/gym-gazebo at master · RoboticsLabURJC/2019-tfm-ignacio-arranz · GitHub*. (s. f.). Recuperado 8 de julio de 2024, de https://github.com/RoboticsLabURJC/2019-tfm-ignacio-arranz/tree/master/gym-gazebo

[23] *Gym-gazebo/INSTALL.md at master · erlerobot/gym-gazebo*. (s. f.). GitHub. Recuperado 8 de julio de 2024, de https://github.com/erlerobot/gym-gazebo/blob/master/INSTALL.md