ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

INGENIERÍA EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA MULTI AGENTE PARA COMUNICAR CONOCIMIENTOS DE APRENDIZAJE POR REFUERZO Y MEJORAR EL COMPORTAMIENTO DE LOS AGENTES

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

DAVID ALEXANDER CÁRDENAS GUILCAPI

david.cardenas@epn.edu.ec

Director: Henry Patricio Paz Arias

henry.paz@epn.edu.ec

2020

AVAL

Como director del trabajo de titulación “Diseño e implementación de una arquitectura multi agente para comunicar conocimientos de aprendizaje por refuerzo y mejorar el comportamiento de los agentes” desarrollado por David Alexander Cárdenas Guilcapi, estudiante de la carrera de ingeniería en sistemas informáticos y de computación de la facultad de ingeniería de sistemas, habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la Defensa oral.

Henry Patricio Paz Arias

DIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, David Alexander Cárdenas Guilcapi, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

David Alexander Cárdenas Guilcapi

DEDICATORIA

Dedico a este trabajo a mi sobrina Romina, pequeña aún no te conozco, pero sé que te adoro con mi vida.

AGRADECIMIENTO

Quiero agradecer a mis padres José y Carmen, gracias por todo lo que han hecho por mí y Doris. Tengan la seguridad que todas las horas de trabajo y sacrificio de su parte han sido enormemente apreciadas, tengan la seguridad de que haré mi mejor esfuerzo por recompensarlos.

A mi hermana Doris, porque siempre pude contar con ella. Gracias también por consentirme y hacerme sentir mejor cuando más lo necesitaba. Ahora es mi turno de consentir a su pequeña.

También gracias a Aldo por cada palabra de apoyo y las veces que con Doris me salvaron de diversas situaciones.

A Diana, mi mejor amiga, por todo su apoyo incondicional, sin duda la mejor persona que he tenido el gusto de conocer.

ÍNDICE DE CONTENIDO

RESUMEN

El presente trabajo presenta una arquitectura multi agente que utiliza aprendizaje por refuerzo. El objetivo es diseñar un sistema que permita a los agentes aprovechar el conocimiento de sus similares. El conocimiento del ambiente es obtenido a partir del algoritmo de aprendizaje por refuerzo, Q-*learning*. La arquitectura multi agente establece un modelo de comunicación entre los agentes del sistema. Para alcanzar el objetivo, el presente trabajo de investigación aprovecha la característica *off-policy* de Q-*learning* incorporando una condición antes de utilizar ε-*greedy*. Esta condición permite que los agentes no exploren un estado que ya ha sido comunicado por otro o por el mismo agente. En la arquitectura multi agente propuesta los agentes trabajan en pares. Cada par de agentes representa dos comportamientos diferentes que les permiten comunicar y operar sobre estados relevantes del ambiente. Las condiciones para comunicar los estados varían dependiendo del ambiente, específicamente, dependen de las circunstancias en las que el agente obtiene una recompensa del ambiente. Los resultados del presente trabajo de titulación muestran que, utilizando la arquitectura propuesta, el número de interacciones agente-ambiente que requieren los agentes para mejorar su comportamiento se reduce en más de un 90%.

Palabras clave: Sistema multi agente, Aprendizaje por refuerzo, Q-*learning, epsilon-greedy.*

ABSTRACT

This research project presents a multi agent architecture which uses reinforcement learning. The goal is to design a system that able the agents take advantage of its peers' knowledge. The knowledge of the environment is obtained from the reinforcement learning algorithm, Q-learning. While, the multi agent architecture sets a communication model between the agents of the system. To reach the goal, the present research project takes advantage of the Q-learning characteristic, off-policy, incorporating a condition before the use of ε-*greedy*. This condition allows the agents not to explore a state that has already been sent by another agent, or itself. In the proposed multi agent architecture the agents work in pairs. Each pair of agents have two different behaviors allowing them to communicate and work on relevant states of the environment. The conditions to send the states depend on the environment, specifically, it depends on the circumstances which the agent obtains a reward from the environment. The results of this work show that through the proposed architecture, the number agent-environment interactions that the agents need to improve their behavior is reduced by more than 90%.

Keywords: Multi agent system, reinforcement learning, Q-learning, epsilon-*greedy.*

DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA MULTI AGENTE PARA COMUNICAR CONOCIMIENTOS DE APRENDIZAJE POR REFUERZO Y MEJORAR EL COMPORTAMIENTO DE LOS AGENTES

1. INTRODUCCIÓN

El presente trabajo de titulación involucra dos temas de Inteligencia Artificial, sistemas multi agente y aprendizaje por refuerzo. Los sistemas multi agente están compuestos de múltiples elementos informáticos interactivos, conocidos como agentes. Un agente es un sistema informático que posee la capacidad de tomar acciones de manera autónoma y de interactuar con otros agentes [1]. Los sistemas multi agente son considerados también una herramienta, que permiten entender y construir un amplio rango de sistemas sociales artificiales.

Aprendizaje por refuerzo se enfoca más en el aprendizaje orientado hacia objetivos a partir de la interacción que otros enfoques de aprendizaje automático [2]. El aprendizaje por refuerzo se ha convertido en un gran campo de investigación con cientos de investigadores alrededor del mundo en diversas disciplinas como psicología, teoría de control, inteligencia artificial y neurociencia [2]. Aprendizaje por refuerzo intenta maximizar una señal de recompensa. De hecho, es similar a lo que describe una teoría dentro del campo de la neurociencia, que establece que las células de dopamina proveen una “señal de predicción de error", lo que permite a los humanos aprender por refuerzo [3].

Como motivación para el presente trabajo de titulación se encuentra el proyecto de Fusté et al [4]. Se trata de un proyecto de realidad aumentada en donde los agentes tienen como objetivo evitar obstáculos que se encuentran dentro de una mesa y a su vez, no caer de ella. Cuando un agente choca con un obstáculo o cae de la mesa, se crea uno nuevo al cual se le asigna un padre y una madre que ya se encuentran en el ambiente. El agente recién creado aprende en el momento en el que los agentes padre y madre chocan con un obstáculo o caen de la mesa. [4].

El comportamiento observado en el trabajo de Fusté puede cambiar, de manera que el agente no espere a que otros cometan equivocaciones (se sacrifiquen) para poder aprender sobre el ambiente. Con el fin de acercar el comportamiento de los agentes a uno más similar al humano, es decir, aprender de las interacciones propias y de las de sus similares, se utiliza aprendizaje por refuerzo junto a un enfoque multi agente.

En cuanto al aprendizaje por refuerzo multi agente se plantea el paradigma del planeamiento centralizado y ejecución descentralizada. El planeamiento centralizado hace referencia al uso de un solo algoritmo para todos los agentes. La ejecución descentralizada se refiere a que cada agente recibe observaciones diferentes lo que provoca que tome acciones diferentes. Cuando un agente ejecuta la política aprendida, no conoce sobre las acciones y estados de otros agentes [5]. Entre los problemas encontrados dentro del enfoque centralizado se encuentra el problema del agente perezoso [6].

Dentro del presente trabajo, los agentes buscan un mismo objetivo que consiste en evitar cometer equivocaciones dentro del ambiente. Se trata de un problema de cooperación. Bajo este contexto, un grupo de agentes puede llegar a una situación de estancamiento, el cual puede ser provocado por el ambiente o por dilemas sociales entre agentes [7].

El presente trabajo aborda el aprendizaje por refuerzo multi agente mediante una arquitectura en donde los agentes trabajan en pares. El objetivo es mostrar que un agente puede cambiar su comportamiento en base a las influencias de sus similares. Este modo de actuar es observado en los humanos, ya que las personas tendemos a hacer lo que aquellos a nuestro alrededor están haciendo [8]. El modelo de aprendizaje de refuerzo le permitirá al agente aprender sobre el ambiente y la arquitectura multi agente le permitirá interactuar con sus similares. La integración de estas dos técnicas permitirá al agente intercambiar mensajes que contengan información relevante, la cual es obtenida a partir de su interacción con el ambiente. Esta información relevante consta de estados recompensas y acciones llevadas a cabo en el ambiente.

Para poner a los agentes a prueba se diseñan dos ambientes, uno llamado “Mesa” y el segundo llamado “Carrera de obstáculos”. Los objetivos y las recompensas de cada uno de los ambientes son diferentes. En el ambiente “Carrera de obstáculos”, el agente deberá llegar a una meta fija en el ambiente en el menor número de movimientos. Par

a lograrlo, el agente debe evitar los obstáculos que se encuentran por delante de él. En contraposición, en el ambiente “Mesa” el agente deberá evitar sobrepasar un margen delimitado, es decir, evitar caer de la mesa. El fin es demostrar que, en base a la colaboración de los agentes, el agente puede aprender más rápido sobre el ambiente y por lo tanto disminuir el número de equivocaciones dentro del mismo. Las observaciones que el agente percibe son posiciones y distancias relativas, lo cual no demanda una gran capacidad de procesamiento computacional.

En el segundo apartado de este documento se encuentra el fundamento teórico bajo el cual se realizó el trabajo de titulación. El tercer apartado presenta la metodología. En el cuarto apartado se encuentran los resultados y discusión. El quinto apartado contiene las conclusiones y recomendaciones. El sexto apartado contiene las referencias bibliográficas y, en el séptimo apartado se encuentran los anexos.

1.1 Pregunta de investigación

¿Puede un agente inteligente aprender sobre el ambiente en el que se encuentra a través de la observación directa del comportamiento de sus similares?

1.2 Objetivo general

Diseñar e implementar una arquitectura multi agente capaz de comunicar conocimientos de aprendizaje por refuerzo para mejorar el comportamiento de los agentes.

1.3 Objetivos específicos

- Diseñar una arquitectura multi agente para lograr que todos los agentes se comuniquen entre sí, dentro de un ambiente simulado.
- Implementar un modelo de aprendizaje por refuerzo de tipo *off-policy* para entrenar al agente a evitar obstáculos dentro del ambiente.
- Integrar el modelo de aprendizaje por refuerzo de tipo off-policy al comportamiento de los agentes con el fin de comunicar la recompensa obtenida, para evitar que los agentes terminen su ciclo de vida dentro del ambiente.

1.4 Hipótesis

Un agente inteligente puede aprender sobre el ambiente en el que se encuentra a través de la observación directa del comportamiento de sus similares, permitiéndole aumentar su conocimiento sin necesidad de cometer directamente una equivocación, con el fin de comportarse de manera óptima dentro de dicho ambiente. Un comportamiento óptimo se refiere a no cometer equivocaciones dentro del ambiente.

2. MARCO TEÓRICO

2.1 Sistemas Multi agente

Los sistemas multi agente son sistemas compuestos de múltiples agentes. Un agente inteligente es un sistema informático que posee dos importantes características. Primero, son al menos hasta cierto punto capaces de actuar de manera autónoma, de decidir por sí mismos lo que necesitan hacer para satisfacer sus objetivos propuestos. Su actuación se limita al dominio en el que actúan. Segundo, son capaces de interactuar con otros agentes, no solamente intercambiando datos, sino también participando dentro de ciertos tipos de actividad social como: cooperación, coordinación, negociación, y similares [1].

La figura 1 [9] muestra la estructura típica de un sistema multi agente. El sistema contiene un número de agentes que interactúan entre sí. Cada agente posee una “esfera de influencia”, lo que significa que tendrá control, o al menos influirá, sobre diferentes partes del ambiente. Las esferas de influencia pueden coincidir en algunos casos, solaparse y producir la aparición de relaciones de dependencia, provocando que no sea posible para ellos alcanzar un objetivo de manera simultánea. Finalmente, los agentes pueden estar enlazados por otro tipo de relación como las de "poder" en donde un agente manda sobre otro.

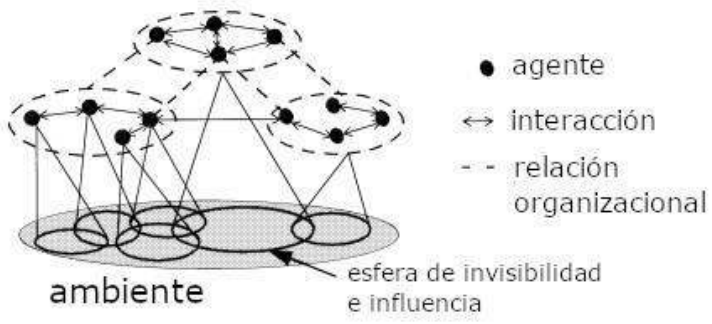


Figura 1. Vista canónica de un sistema basado en agentes.

Entre los beneficios de la utilización de un sistema multi agente [10] se encuentran:

- Aceleración y eficiencia computacional dentro del proceso debido a la utilización de computación paralela.
- Confiabilidad y robustez en el contexto de que si un agente falla, el sistema puede sobrellevar esta falla.
- Escalabilidad y flexibilidad debido a que es sencillo agregar nuevos agentes al sistema.
- Menor costo de mantenimiento ya que es más sencillo mantener un sistema de agentes debido a su modularidad.
- Reusabilidad, es más fácil mantener a un agente que a un sistema monolítico.

Uno de los desafíos dentro de un sistema multi agente es la percepción [11]. Dado que los agentes se encuentran repartidos por todo el ambiente, cada agente tiene una capacidad limitada para obtener información del mismo. Esta capacidad es limitada por la cobertura de los sensores a los que se encuentran conectados los agentes. De igual manera, un sistema puede poseer agentes con diferentes objetivos y capacidades. Los agentes de diferente comportamiento pueden no cooperar ni competir entre ellos. Por tal razón, es crucial encontrar un mecanismo de inferencia que se adapte a las capacidades de cada agente.

2.2 Aprendizaje por refuerzo

Aprendizaje por refuerzo se refiere a aprender qué hacer [2]. Las características más importantes del aprendizaje por refuerzo son la prueba y el error y la recompensa retrasada. La primera se relaciona con el hecho de que al agente no se le indica qué acciones tomar, en su lugar, tiene que descubrir cuáles son las acciones que producen la mayor recompensa tras interactuar con el ambiente. La recompensa retrasada se refiere a que en ciertos casos las acciones no solamente pueden afectar a la recompensa inmediata, sino también la siguiente

situación y, a través de ello, todas las recompensas subsecuentes. El problema del aprendizaje por refuerzo se formaliza como el control óptimo de un proceso de decisión de Markov.

El aprendizaje por refuerzo es diferente al aprendizaje supervisado; en el aprendizaje supervisado el sistema busca generalizar sus respuestas de manera que logre actuar correctamente en situaciones que no están presentes en el conjunto de entrenamiento. A pesar de que el aprendizaje supervisado es un importante tipo de aprendizaje, por sí solo, no es adecuado para el aprendizaje a partir de la interacción. El aprendizaje por refuerzo también se diferencia del aprendizaje no supervisado, ya que el aprendizaje por refuerzo busca maximizar una recompensa en lugar de intentar encontrar una estructura oculta. Por lo tanto, según Sutton y Barto [2], se considera al aprendizaje por refuerzo como un tercer paradigma del *machine learning,* junto al aprendizaje supervisado y aprendizaje no supervisado.

Una de las cuestiones más importantes en el aprendizaje por refuerzo es balancear la explotación y la exploración [12]. Exploración se refiere a continuamente intentar nuevas maneras de resolver el problema, mientras que la explotación busca capitalizar soluciones que ya se encuentran establecidas. Las soluciones que son consideradas como buenas pueden deteriorarse con el tiempo, o pueden aparecer mejores soluciones. Estas mejores soluciones no podrían llegar a conocerse sin la exploración, lo que eventualmente causará que el sistema se deteriore.

Todos los agentes de aprendizaje por refuerzo tienen metas explícitas, pueden obtener información sobre los aspectos de su ambiente, y elegir acciones para influir en sus ambientes. Además, el agente tiene que actuar a pesar de la incertidumbre que existe sobre el ambiente que enfrenta.

Los aspectos más emocionantes del aprendizaje por refuerzo moderno son sus fructíferas interacciones con otras ingenierías y disciplinas científicas. De todas las formas de *machine learning* el aprendizaje por refuerzo es el más cercano al tipo de aprendizaje que humanos y otros animales realizan, y varios de los algoritmos principales del aprendizaje por refuerzo fueron inspirados originalmente por sistemas de aprendizaje biológico [2 pp. 3-4].

2.3 Proceso de decisión de Markov

Dentro de aprendizaje por refuerzo es fundamental la interacción entre el agente y el ambiente. Esta interacción es descrita formalmente por el proceso de decisión de Markov (MDP, por sus siglas en inglés) y se observa en la figura 2 [13]. En esta figura se muestra que un agente realiza una acción en el ambiente, esto provoca que el ambiente se actualice al siguiente estado y presente una recompensa al agente.

Figura 2. La interacción agente-ambiente en un proceso de decisión de Markov.

Específicamente, el agente y el ambiente interactúan en cada uno de una secuencia de intervalos de tiempo discretos $t = 0,1,2,3 \ldots$ En cada t, el agente recibe cierta representación del estado del ambiente $S_t \in \mathcal{S}$ (conjunto de estados) y sobre ello selecciona una acción $A_t \in \mathcal{A}(s)$ (conjunto de acciones pertenecientes a un estado). Un intervalo de tiempo después, en parte como consecuencia de su acción, el agente recibe una recompensa, $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$, siendo $\mathcal{R}$ el conjunto de recompensas disponibles, y se encuentra a sí mismo en un nuevo estado, S_{t+1}. El MDP y el agente dan origen a una secuencia, conocida como trayectoria, que se expresa en la ecuación 1 [2 pp. 48].

$$S_0, A_0, R_0, S_1, A_2, R_2, S_3, A_3, R_3 \ldots\ldots$$

Ecuación 1. Trayectoria dentro de un MDP

En un MDP finito, los conjuntos de estados, acciones y recompensas tienen un número finito de elementos. Las recompensas y estados tienen una distribución de probabilidad discreta bien definida que dependen solamente del estado y acción precedentes. Para valores particulares de estas variables aleatorias, $s_t' \in \mathcal{S}$ y $r \in \mathcal{R}$ existe una probabilidad de que estos valores ocurran a un tiempo t, dados valores particulares del estado y acción precedentes. La función p (Ecuación 2) define las dinámicas del MDP para todo $s_t', s \in \mathcal{S}, r \in \mathcal{R}$ y $a \in \mathcal{A}\, s_t$.

$$p\ s_t', r_t s, a_t \doteq \Pr\{S_t = s_t', R_t = r \mid S_{t-1} = s, A_t - 1 = a\}$$

Ecuación 2. Dinámicas del MDP [2 pp.48].

Dentro de un MDP existe una probabilidad de transición la cual determina cómo el sistema se moverá hacia el siguiente estado [14]. La ecuación 3 indica que cada valor posible para S_t y R_t depende solamente del estado y acción S_{t-1} y A_{t-1} inmediatamente precedentes. El estado debe incluir información sobre todos los aspectos de la interacción agente-ambiente anterior que marcan la diferencia para el futuro. Si lo hace, entonces el estado cumple con la propiedad de Markov [2 pp. 49].

$$p\ s'\!,r\!s,a\! \doteq \Pr\{S\!t\! =\ s'\! \mid S\!t-1\! = s,\ A\!t-1\! = a\} =\ r \in \mathcal{R}\!\!p\ s'\!,r\!s,a\!\!$$

Ecuación 3. Probabilidades de transición de estado.

Dentro de aprendizaje por refuerzo, existe una recompensa la cual determina la consecuencia inmediata para la acción a elegida por el agente en un estado s [14 pp. 2]. El valor de la recompensa depende del siguiente estado del sistema convirtiéndose en una recompensa esperada. Esta recompensa esperada se expresa en la ecuación 4.

$$r(s,a) \doteq\ \mathbb{E}[\,R\!t\! \mid S\!t-1\! = s,\ A\!t\!-1 = a] =\ r \in \mathcal{R}\!\!r\ \ s'\! \in \mathcal{S}\!\!p\ s'\!,r\!s,a\!\!\!$$

Ecuación 4. Recompensas esperadas para los pares estado-acción [2 pp. 49]

Esta recompensa pasa del ambiente al agente en cada intervalo de tiempo, la recompensa es un número $R\!t\! \in\ \mathbb{R}$. El objetivo del agente es maximizar la cantidad total de recompensa que recibe, es decir, maximizar no solo la recompensa inmediata sino la recompensa acumulada a largo plazo. Si se quiere alcanzar un objetivo se le debe proveer recompensas al agente, cuando el agente maximice la recompensa acumulada, alcanzará el objetivo. Es crítico que las recompensas verdaderamente indiquen lo que se desea alcanzar. Es decir, no otorgar recompensas por alcanzar submetas ya que puede llevar al agente a buscar una forma de obtener estas submetas y olvidar el objetivo real [2 pp. 54].

Si la secuencia de recompensas recibidas después del intervalo t se denota como $R\!t+1\!,\ R\!t+2\!,\ R\!t+3\!,$ ..., lo que se busca maximizar es el retorno esperado, $G\!t\!$, el cual es definido como cierta función específica de la secuencia de recompensas. En el caso más simple, se muestra al retorno como la suma de las recompensas, como se muestra en la ecuación 5.

$$G\!t\! \doteq\ R\!t+1\! +\ R\!t+2\! +\ \ R\!t+3\! + \cdots +\ \ R\!T\!$$

Ecuación 5. Retorno esperado expresado como la suma de las recompensas [2 pp. 54].

En donde T es un instante final. Este enfoque tiene sentido al tomar en cuenta aplicaciones en donde la interacción agente-ambiente se divide de forma natural en subsecuencias. Estas subsecuencias se denominan episodios. Cada episodio termina en un estado especial llamado el “estado terminal”, seguido de un reinicio a un estado inicial. Un episodio puede terminar de diferentes maneras y esto no influye en cómo inicia el siguiente episodio, esto dado que cada episodio puede alcanzar el estado terminal con diferentes recompensas. Las tareas con episodios se denominan tareas episódicas o modelos de horizonte finito [15].

Cuando la interacción agente-ambiente no se rompe naturalmente en episodios identificables se trata de una “tarea continua”, denominada también como un modelo de horizonte infinito. En una tarea continua se toma en cuenta la recompensa a largo plazo, pero estas recompensas que se reciben en el futuro son descontadas en función a qué tan lejos en el tiempo se reciben. Para esto, se utiliza factor de descuento $0 < \gamma < 1$. Este factor de descuento asegura que incluso al ser un modelo de horizonte infinito, la suma de las recompensas sea una cantidad finita [15].

La tasa o factor de descuento determina el valor presente de las recompensas futuras: una recompensa recibida k instantes de tiempo en el futuro vale solamente γ^{k-1} veces lo que valdría si fuera recibida inmediatamente [2 pp.55]. Si $\gamma = 0$ el agente es considerado “miope”, se enfoca solo en maximizar las recompensas inmediatas. Si las acciones del agente influyen solo en la recompensa inmediata, se reduce el acceso a las recompensas futuras y el retorno es reducido. Mientras que si γ se acerca a 1, el retorno objetivo toma más en consideración las recompensas futuras. La fórmula del retorno descontado esperado se encuentra en la ecuación 6.

$$G(t) \doteq R(t+1) + \gamma R(t+2) + \gamma^2 R(t+3) + \cdots = \sum_{k=0}^{\infty} \gamma^k R(t+k+1)$$

Ecuación 6. Retorno descontado esperado [2 pp. 54].

El algoritmo de un agente dentro de aprendizaje por refuerzo para escoger a en un estado actual s, se expresa como la distribución $\pi(a|s)$ y se denomina política (estrategia) [16].

Para un MDP y política π existe una función de valor bajo una política π , la cual se encuentra definida en la ecuación 7.

$$v_\pi(s) \doteq \mathbb{E}[G(t) \mid S(t) = s] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(t+k+1) \mid S(t) = s\right], \text{ para todo } s \in \mathcal{S}$$

Ecuación 7. Función de valor bajo una política π [2 pp.58]

Para un MDP finito, una política π es mejor o igual que una política π' si su retorno esperado es mayor o igual que la de π' para todos los estados. En otras palabras $\pi > \pi'$, si y solo si $v_\pi(s) > v_{\pi'}(s)$ para todo , $s \in \mathcal{S}$. Siempre existe al menos una política que es mejor o igual que las demás políticas, a esto se le conoce como una política óptima [2 pp.62].

La función de valor estima qué tan bueno es para el agente utilizar la estrategia π para visitar el estado s y generaliza la noción del retorno descontado esperado que corresponde a $v_\pi(s0)$. Dado que una función de valor puede obtenerse a partir de cualquier política, si

se toma en cuenta la política óptima, se obtiene la función de valor óptima que se define en la ecuación 8.

$$v_*(s) \doteq \max_\pi v_\pi(s)$$

Ecuación 8. Función estado-valor óptima [2 pp. 62]

Obtener $v_*(s)$ no provee la suficiente información para reconstruir la política óptima π_* debido a que se desconocen ciertas dinámicas del ambiente como por ejemplo las probabilidades de transición (Ecuación 3). La función de valor no es útil si el agente no conoce qué estado puede aparecer en el ambiente como respuesta a una acción. Por este motivo se introduce una nueva noción con más información denominada la función acción valor o función de calidad (*quality function*, Q-*function*). Se define en la ecuación 9.

$$q_\pi(s,a) \doteq \mathbb{E}[G_t \mid S_t = s, A_t = a] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

Ecuación 9. Función acción-valor para la política π [2 pp.58]

Para una política óptima se encuentra $q_*(s,a)$, que se describe en la ecuación 13.

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a)$$

Ecuación 10. Función acción-valor óptima [2 pp. 63.]

Obtener la función Q óptima significa resolver una tarea de aprendizaje por refuerzo [16].

2.4 Diferencia Temporal y Q-Learning

El aprendizaje por diferencia temporal (TD, por sus siglas en inglés) es una combinación de las ideas de Monte Carlo y programación dinámica [2 pp. 119]. Los métodos TD pueden aprender directamente de la experiencia sin tener un modelo de las dinámicas del ambiente. Dada alguna experiencia al seguir una política π, los métodos TD actualizan su estimación V de v_π para los estados no terminales S_t que ocurren en esa experiencia. Los métodos TD realizan una actualización útil en el momento $t+1$, utilizando la recompensa R_{t+1} y la estimación $V(S_{t+1})$. Esto marca una clara diferencia con los métodos Monte Carlo en donde se necesita esperar hasta el final del episodio para realizar un incremento de $V(S_t)$. El método TD más simple realiza la actualización descrita en la ecuación 11:

$$V(S_t) \leftarrow V(S_t) + \alpha\left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]$$

Ecuación 11. Actualización del método TD [2 pp. 119].

Q-learning es un algoritmo de control de diferencia temporal de tipo *off-policy,* cuya actualización se describe en la ecuación 12:

$$Q(S\square t\square, A\square t\square) \leftarrow Q(S\square t\square, A\square t\square) + \alpha\ R\square t+1\square + \gamma\ \max\square a\square\square\ Q(S\square t+1\square, a)\square - Q(S\square t\square, A\square t\square)\ \square$$

Ecuación 12. Actualización del algoritmo Q-learning [2 pp. 131]

Se trata de un algoritmo *off-policy* dado que evalúa o mejora una política diferente de la que se utiliza para generar los datos [2 pp. 100]. Es decir, el aprendizaje de la función de valor de una política objetivo proviene de datos generados a partir de una política diferente llamada política de comportamiento. En Q-*learning*, la función de acción-valor aprendida, Q, aproxima directamente $q\square *\square$, la función acción-valor óptima. Esta aproximación se realiza independientemente de la política que está siguiendo, aunque esta aún tiene efecto determinando qué pares de estado-acción son visitados y actualizados [2 pp. 131]. El algoritmo Q-*learning* se describe en la figura 3.

Parámetros del algoritmo: taza de aprendizaje $\alpha \in (0,1]$, ε pequeño >0

Inicializar $Q(s,a)$, para todos los $s \in S^+, a \in A(s)$, arbitrariamente.

Bucle para cada episodio:

Inicializar S

Bucle para cada paso del episodio:

Elegir una acción A de las pertenecientes al estado S utilizando la política derivada de Q (por ejemplo, $\varepsilon - greedy$)

Tomar la acción A, observar R, S'

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R_{t+1} + \gamma \max_a Q(S',A) - Q(S,A)\right]$$

$$S \leftarrow S'$$

Hasta que S sea terminal

Figura 3. Algoritmo Q-learning [17]

2.5 Epsilon-greedy

Epsilon-*greedy*, ε-*greedy*, es un método utilizado para balancear la exploración y explotación Con ε-*greedy*, el agente selecciona en cada intervalo de tiempo una acción aleatoria con una probabilidad establecida $0 \leq \varepsilon \geq 1$, en lugar de seleccionar de manera voraz una de las acciones óptimas con relación a la función Q. ε-*greedy* expresado de otra forma indica:

$$\pi(s)\ \ acción\ aleatoria\ obtenida\ de\ A\ s\square \qquad si\ \mathfrak{E} < \varepsilon\ \square arg\ \max\square a\square\square Q\ s,a\square\ \in A\ s\square\square \qquad caso\ contrario\square\square$$

En donde $0 \leq\ \mathfrak{E}\ \geq 1$ es un número obtenido en cada intervalo de tiempo [18].

3. METODOLOGÍA

3.1 Diseño de la arquitectura multi agente

Para el desarrollo de la arquitectura se tomó en cuenta el planeamiento centralizado y ejecución descentralizada, el cual es considerado un paradigma dentro del planeamiento multi agente [19], [20].

La arquitectura propuesta se observa en la figura 4, en donde, se estableció que los agentes trabajen en pares, cada par de agentes está conformado de un agente denominado “Explorador” y uno denominado “Almacén”. Todos los agentes de tipo “explorador” utilizaron el mismo algoritmo y lo mismo sucedió con todos los agentes de tipo “almacén”.

Los agentes “almacén” y “explorador” se diferenciaron por sus comportamientos. El agente "explorador” se encargó de interactuar con el ambiente, obtener información del mismo y modificar su política, mientras que el agente “almacén” acumuló la información que los agentes "explorador” le enviaron. Cuando el agente de tipo explorador interactuó con el ambiente, sus similares no conocieron sus estados o recompensas. De igual manera, los agentes de tipo almacén, no conocieron la información que poseían los otros agentes “almacén”.

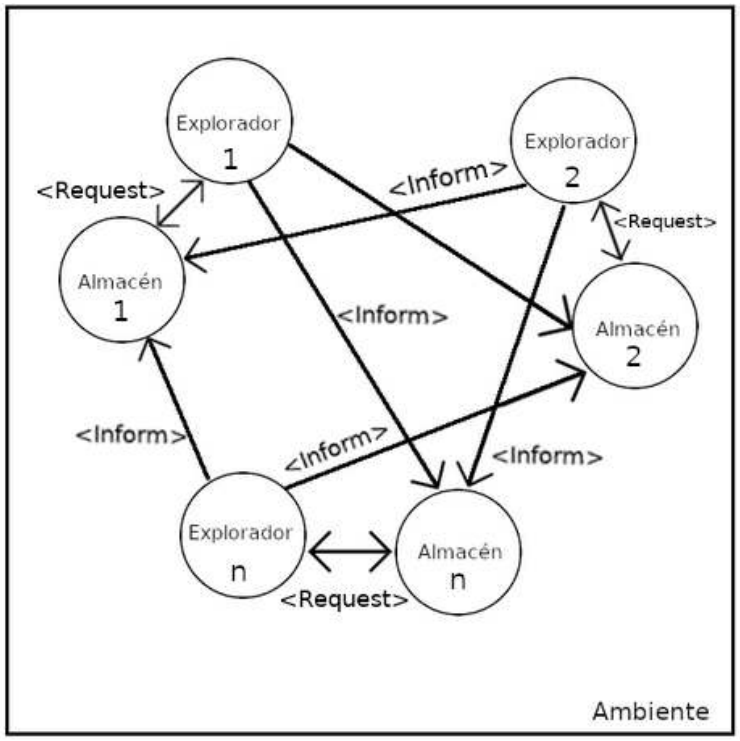


Figura 4. Arquitectura multi agente propuesta.

Cada agente de tipo explorador solicitó información a su pareja de tipo almacén al inicio de cada episodio. Dado que los agentes se ejecutaron en paralelo, cualquiera de los agentes del

sistema pudo ser el primero en hacer contacto con su pareja. El agente de tipo “explorador” fue diseñado para enviar información a todos los agentes de tipo “almacén” del sistema, por lo contrario, el agente “almacén” fue diseñado con el propósito de contactarse únicamente con su pareja “explorador”. Las parejas de agentes son visibles mediante sus nombres, las parejas de agentes se identificaron por el mismo número al final de su nombre.

De acuerdo con Dorri [21], el sistema multi agente posee las características descritas en la tabla 1.

Criterio	**Característica del sistema multi agente**	**Descripción**
Liderazgo	Sin líder	Cada agente decide sus acciones de manera autónoma.
Función de decisión	Lineal	La decisión de un agente es proporcional a los parámetros obtenidos del ambiente.
Heterogeneidad	Heterogéneo	Existen agentes con características diferentes.
Parámetros de acuerdo	Primer orden	Los agentes colaboran para acordar una sola métrica.
Consideración de retraso	Sin retraso	Los retrasos no son substanciales o relevantes.
Topología	Estático	Las relaciones de los agentes no cambian a lo largo del ciclo de vida del agente.
Frecuencia de la transmisión de datos	Activado por eventos	El agente envía los datos cuando ocurre un evento.
Movilidad	Dinámico	El agente se mueve a través del ambiente.

Tabla 1. Características del Sistema multi agente

3.2 Diseño de los modelos de aprendizaje por refuerzo

El algoritmo de aprendizaje por refuerzo seleccionado fue Q-learning por su característica de off-policy. En cuanto a la política de comportamiento se seleccionó ε-greedy ya que al momento de explorar permite seleccionar una acción de manera aleatoria. Se implementó también un decaimiento de ε de tipo exponencial que se produjo al final de cada episodio. ε tuvo un valor inicial de 1 y el decaimiento fue fijado en 0.9998. Este decaimiento se implementó teniendo en cuenta que la exploración es más importante cuando el agente no

tiene la suficiente información para interactuar correctamente con el ambiente. Una vez que el agente obtiene más información, el decaimiento permite al agente explotar más su conocimiento [18]. Al momento de poner a prueba las políticas, no se utiliza el decaimiento de ε.

Se diseñaron dos ambientes, los objetivos del agente son diferentes en cada uno de ellos. Las recompensas fueron diseñadas en base a los objetivos del agente, por lo tanto, son propias de cada ambiente. Los ambientes son denominados como: “Mesa” y “Carrera de obstáculos” y según Wooldridge [1] poseen las características descritas en la tabla 2.

Criterio	**Característica de los ambientes**	**Descripción**
Accesibilidad	Accesibles	Los agentes pueden obtener información completa, precisa y actualizada del ambiente.
Determinismo	Determinísticos	Dentro del ambiente cada acción del agente tiene un efecto conocido.
Dinamismo	Estáticos	El ambiente no sufre cambios a excepción de los provocados por las acciones del agente.
Continuidad	Discretos	Existen un número finito y establecido de acciones y estados.

Tabla 2. Características de los ambientes "Mesa" y "Carrera de obstáculos".

Ambiente “Mesa”

Este primer ambiente fue inspirado por el trabajo de Fusté [3]. En el trabajo mencionado los agentes se encuentran sobre una mesa y su objetivo es aprender a no caer de la misma. En el mencionado trabajo, un agente al caer de la mesa desaparece del ambiente y se le asigna un agente padre y madre, a partir de los cuales aprenderá sobre el ambiente. Este proceso implica que dicho agente necesitará de su padre y madre para aprender del ambiente, es decir, necesita que sus padres caigan de la mesa. Al plantear este ambiente dentro del presente trabajo, se buscó que el agente pueda aprender de sí mismo y de sus similares a no sobrepasar los límites de la mesa. Además, se definió una relación no jerárquica entre los agentes, es decir, todos aprenden de todos.

Para el ambiente “mesa” se diseñó una matriz de tamaño 21x21. La representación del borde de la mesa se marcó a 2 unidades a partir de los filos de la matriz, se representa en color rojo

en la figura 5. Durante el planeamiento, la posición inicial del agente era cualquiera dentro de la zona representada por color azul dentro de la figura 5. El tamaño de esta área se fijó con el fin de que los agentes exploraran la mayor cantidad de estados posibles

Dentro del ambiente “Mesa”, la posición en x y en y del agente representó el estado. Se creó un sistema de coordenadas cuyo origen se colocó en la esquina inferior izquierda de la matriz. Mientras que, el espacio de acciones se limitó a 4:

- Moverse un espacio hacia la derecha.
- Moverse un espacio hacia la izquierda.
- Moverse un espacio hacia abajo.
- Moverse un espacio hacia arriba.

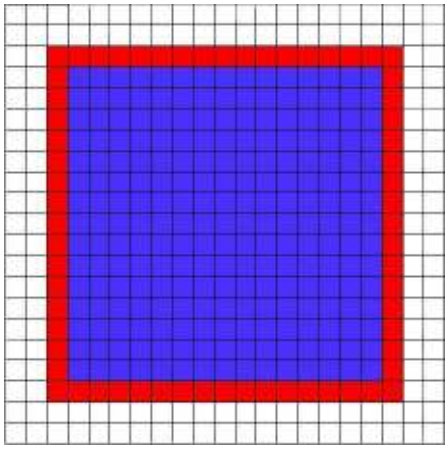


Figura 5. Representación del ambiente "Mesa" y la zona en la que los agentes aparecen durante el planeamiento.

El número máximo de movimientos se fijó en 16, representa el número de unidades entre el límite de la zona en las que aparecen los agentes durante el planeamiento y el borde más lejano. Se diseñó que el episodio llegue a su fin cuando el agente alcance el límite de movimientos o sobrepase el borde.

Se estableció que la recompensa por un movimiento que no sobrepasa el borde tenga un valor inicial de 100. A partir de este valor, la recompensa obtenida por el agente se descontó en función de la distancia al borde más cercano. Un movimiento que sobrepasa el borde es observado en la figura 6. Es decir, mientras más pequeña era la distancia entre el agente desde adentro de la mesa a uno de los bordes, mayor era la recompensa. Para esto, tras cada movimiento se calculó la distancia del agente a cada uno de los bordes (Db) y se tomó la menor de ellas como se describe en la Ecuación 13.

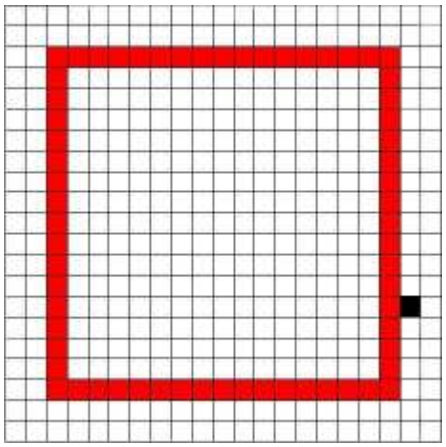


Figura 6. Ejemplo de agente sobrepasando el borde la mesa en el ambiente "Mesa"

$$Distancia\ al\ \ borde[i] = \ posición\ agente - \ borde[i], para\ todos\ los\ bordes, i = 4$$

$$r\ = \ 100/(min(distancia\ al\ borde) + 1\ )$$

Ecuación 13. Recompensa por un movimiento que no sobrepasa el borde.

Las distancias del agente a los bordes superior e inferior se calcularon con la posición del agente sobre el eje y. En cambio, las distancias del agente a los bordes derecho e izquierdo se calcularon con las posiciones del agente sobre el eje x.

Al manejar una recompensa constante, el agente cumplía con su objetivo de mantenerse en la mesa, pero debido a que todos sus movimientos tenían exactamente el mismo valor, el agente se mantenía en una zona central de la mesa. En cambio, el uso de la ecuación 13 causó que la recompensa no fuera constante provocando que el agente se mueva hacia los bordes de la mesa.

Los agentes recibieron una recompensa negativa al sobrepasar el borde. Esta recompensa por caída (Rpc) fue dada en función del número de movimientos máximo que el agente puede tomar y el valor inicial de la recompensa por movimiento por -1, es decir –100, como se indica en la Ecuación 14.

$$Recompensa\ por\ caída\ = -100\ *\ número\ máximo\ de\ movimientos.$$

Ecuación 14. Recompensa por caída del ambiente "Mesa".

Esta recompensa se obtuvo de manera experimental, ya que, de esta manera la recompensa al final de un episodio en el cual el agente sobrepasó el borde la mesa, era negativa.

Para poner a prueba las políticas aprendidas se redujo la zona en la que aparecieron los agentes con el propósito de poder visualizar una mayor cantidad de movimientos. Esta zona está representada en color verde en la figura 7.

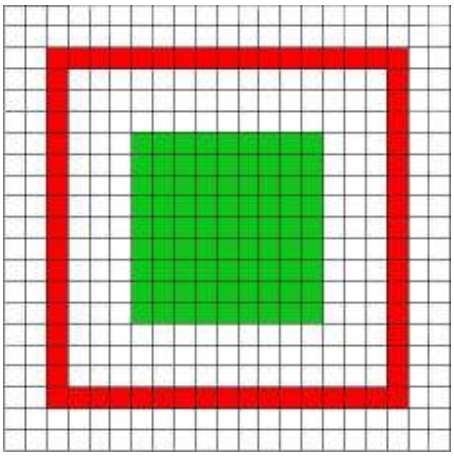


Figura 7. Zona en la que aparecen los agentes durante la ejecución en el ambiente "Mesa".

Ambiente “Carrera de obstáculos”

Este ambiente no fue motivado directamente por otra investigación. El motivo de desarrollar este ambiente se basó en el conocimiento empírico de que, si una persona ve a otra caer, esta persona estará más alerta para no caer también. Para este ambiente se estableció un vector de tamaño 12, en donde se colocaron 3 obstáculos. La posición inicial del agente fue establecida en el extremo izquierdo mientras que la meta fue colocada en el extremo derecho del vector. El ambiente se puede ver en la figura 8 en donde la posición inicial está representada en color azul, los obstáculos en color rojo y la meta en color verde. El agente tuvo como objetivo evitar los obstáculos y alcanzar la meta en el menor número de pasos.

Figura 8. Representación gráfica del ambiente "Carrera de obstáculos"

El estado del agente fue representado por dos distancias:

- Distancia del agente al obstáculo más cercano.
- Distancia del agente a la meta.

Dado que el agente se desplazó sobre solo un eje, estas distancias se calcularon únicamente tomando en cuenta la posición sobre el eje x del agente.

El espacio de acciones para este ambiente es 2:

- Acción salto: moverse 2 espacios hacia la derecha (figura 9).
- Acción paso: moverse 1 espacio hacia la derecha (figura 10).

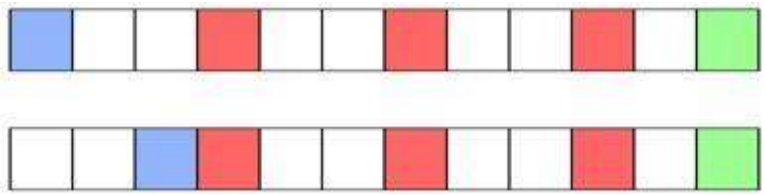


Figura 9. Ejemplo de acción salto.

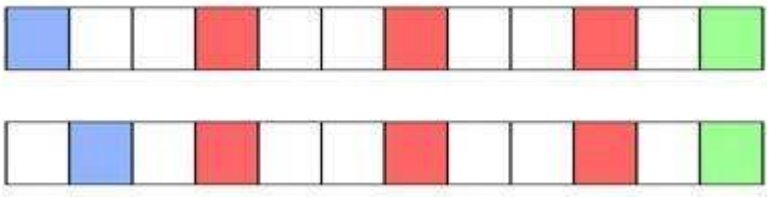


Figura 10. Ejemplo de acción paso

Se estableció que el episodio llegue a su fin si el agente “chocó” con un obstáculo (como se observa en la figura 11, o “alcanzó” la meta. Es decir, cuando la posición del agente coincidió con la posición del obstáculo o de la meta, respectivamente.

Figura 11. Ejemplo de agente "chocando con obstáculo".

El objetivo del agente no fue solamente llegar a la meta, sino hacerlo en el menor número de pasos. Por esta razón, el agente recibió una recompensa de -10. Esto debido a que se consideró mejor para el agente obtener, una recompensa de $x * (-10)$ que $(x + 1) * (-10)$.

En cambio, la recompensa por “chocar” con un obstáculo (Rco), fue definida en función de la posición del obstáculo con el que chocó y la posición de la meta. Se observa en la Ecuación 15.

$$Rco = posición\ de\ la\ meta - posición\ del\ obstáculo$$

Ecuación 15. Recompensa por choque en el ambiente "Carrera de obstáculos"

Esta fórmula se diseñó con el objetivo de que mientras más lejos de la meta el agente “choque”, menor sea su recompensa.

La recompensa por alcanzar la meta se fijó en un valor de +10 por el tamaño del ambiente. Esta recompensa aseguró que la recompensa al final de un episodio en donde un agente alcanzó la meta, sea positiva.

3.3 Integración del aprendizaje por refuerzo dentro de la arquitectura multi agente.

Una vez que se diseñó la arquitectura multi agente y los modelos de aprendizaje por refuerzo, fue necesario que, a través de dicha arquitectura, los agentes comuniquen información obtenida a partir de Q-*learning*.

La figura 12 muestra el proceso de comunicación dentro de la arquitectura, el cual está basado en el intercambio y optimización de estados relevantes del ambiente. Los estados relevantes son aquellos en donde el agente obtuvo una recompensa.

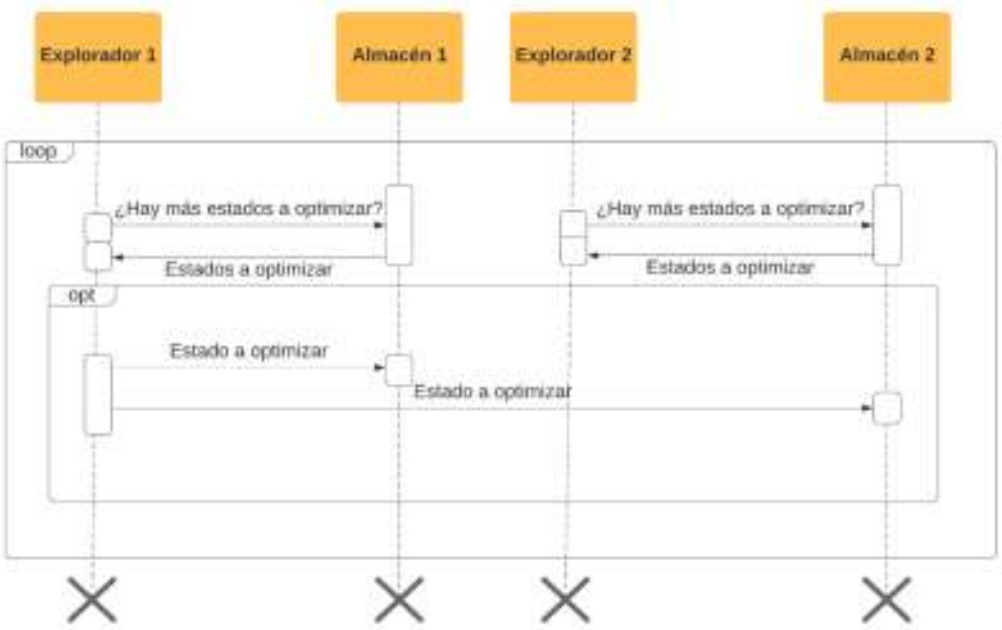


Figura 12. Modelo de comunicación de la arquitectura multi agente propuesta.

En la figura 12 se observa que el agente “explorador” solicita más estados a optimizar. Se determinó que junto a esta solicitud el agente “explorador” envíe también una LEM (Lista de estados modificados), esta es una lista en donde se encuentran los estados que han sido modificados como producto del intercambio de información. El agente almacén responde con una lista de estados a analizar denominada LEAM. Se definió otro momento para enviar información, sucede cuando un agente “explorador” obtiene una recompensa, y envía el estado relevante hacia todos los “almacenes”. Este intercambio entre “exploradores” y “almacenes” se observa en la figura 13.

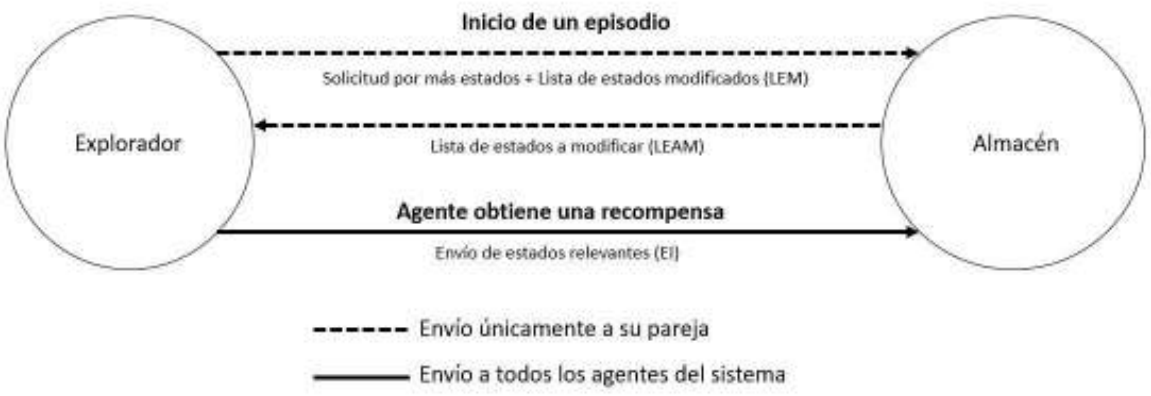


Figura 13. Intercambio de información entre agentes "explorador" y "almacén".

Los denominados estados relevantes son de hecho tuplas, dentro de estas tuplas se encuentran el estado, la acción y el tipo de recompensa. Si bien los estados relevantes son enviados a todos los “almacenes” del sistema, la lista LEM se envía desde el agente “explorador” solamente hacia su pareja.

El comportamiento del agente “almacén”, se diseñó con el fin de acumular la información de todos los agentes “explorador”, pero solamente responder a su pareja. La información que recibió de los agentes de tipo explorador fue acumulada en una lista llamada LER (Lista de estados recibidos), como se observa en la figura 14.

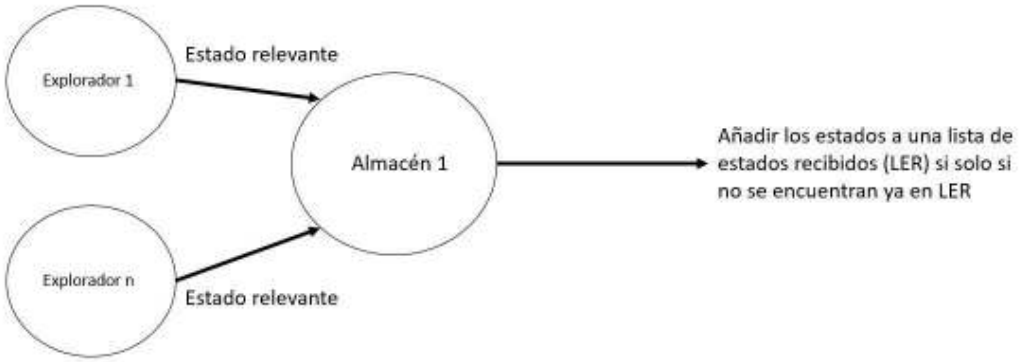


Figura 14. Formación de LER dentro del agente "almacén"

Cuando el agente “almacén” recibió la solicitud de su pareja de tipo explorador, se realizó una comparación entre su LER y la LEM que acaba de recibir para formar la LEAM, (Lista de estados a modificar), como se observa en la figura 15. Esta comparación se realizó entre los estados que conforman las tuplas, ya que LEAM contiene las tuplas cuyos estados se encuentran en LER, pero no en LEM. Este proceso permitió que una vez que se responde la LEAM al agente “explorador”, este analice los estados que enviaron sus similares, pero no más de una vez.

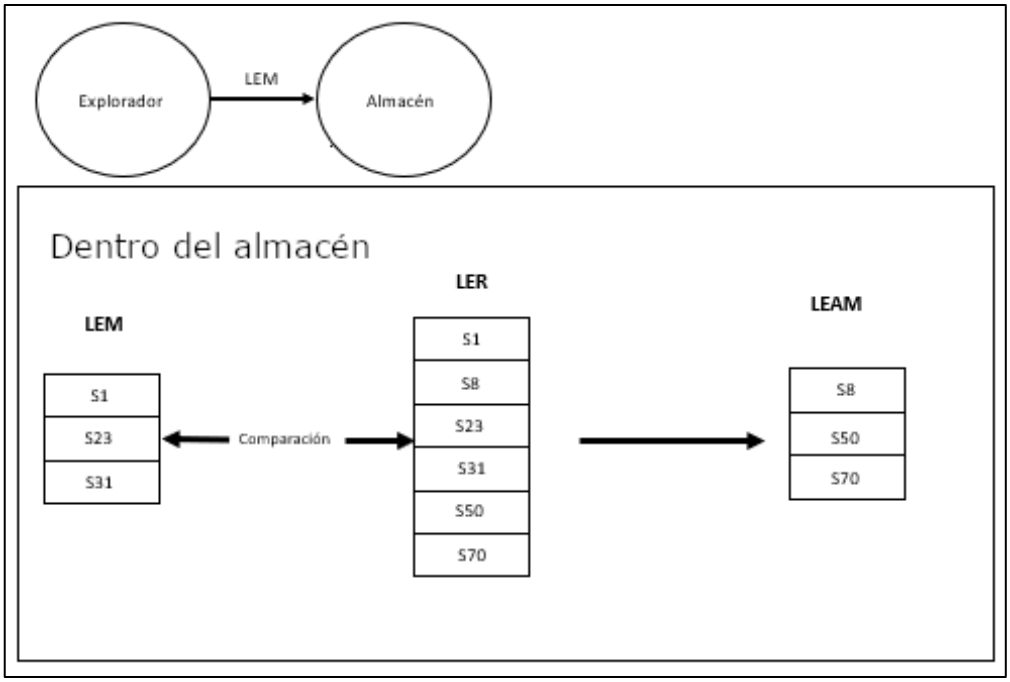


Figura 15. Formación de LEAM dentro del agente "almacén".

El envío de información (EI) observado en la figura 13 ocurre cuando un agente obtiene una recompensa del ambiente. Se estableció que la recompensa pueda ser positiva o negativa, para diferenciarlas se denominó a la recompensa negativa como “penalidad” y a la recompensa positiva simplemente como “recompensa”. Si el agente explorador obtuvo una recompensa positiva, una tupla de la forma: [s, a, “recompensa”] era enviada, si en cambio obtuvo una recompensa negativa, el contenido de la tupla se convirtió en: [s, a, “penalidad”]. Siendo s el estado en donde obtuvo la recompensa, y a la acción que llevó al agente a alcanzar la recompensa o penalidad.

Para mostrar en qué momento se formaron las listas, se agregaron unas líneas al algoritmo de *Q-learning*, las cuales se muestran como pseudocódigo en la figura 16. El proceso diseñado para el agente de tipo explorador muestra que envió la solicitud junto a su LEM al inicio de cada episodio, una vez que recibió la LEAM, analizó los estados de esta y los agregó a LEM.

Parámetros del algoritmo: taza de aprendizaje $\alpha \in (0,1]$, ε pequeño >0

Inicializar $Q(s,a)$, para todos los $s \in S^+, a \in A(s)$, arbitrariamente.

Bucle para cada episodio:

Inicializar S

Bucle para cada paso del episodio:

Solicitar información a par Almacén junto con LEM

Analizar los elementos de LEAM

Revisar LEM

Si S dentro de LEM

Tomar $arg\max_a Q(s,a) \in A(s)$

Sino

Elegir una acción A de las pertenecientes al estado S utilizando la política derivada de Q (por ejemplo, $\varepsilon - greedy$)

Tomar la acción A, observar R, S'

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R_{t+1} + \gamma \max_a Q(S',A) - Q(S,A)\right]$$

$$S \leftarrow S'$$

Hasta que S sea terminal

Figura 16. Modificación del algoritmo Q-learning

El análisis de LEAM se muestra en pseudocódigo en la figura 17 y de manera gráfica en la figura 18. Este análisis está diseñado con el fin de que el agente explorador analice el tercer componente de cada tupla, es decir, el tipo de recompensa. Cuando el tipo de recompensa era “recompensa”, se verificó que la acción que ingresa en la tupla sea la acción de mayor valor. Si al analizar los valores de las acciones, la acción en cuestión no era la de mayor valor, se la convertía en la de mayor valor sumándole una cantidad lo suficientemente grande que fue 100. En cambio, si resultó ser la de mayor valor, solamente se agregó la tupla a LEM. Cuando el tipo de recompensa era “penalidad”, se realizó un proceso similar, es decir, se verificó que la acción que ingresa sea la de menor valor. Si era ya la acción de menor valor se la agregó a LEM directamente, caso contrario, se le sustrajo un valor de 100. Este proceso involucró que se modificaran los valores de las acciones en los estados que el agente “explorador” recibió una recompensa, es decir, se modificó la tabla Q del agente, un ejemplo para los dos tipos de recompensa es observable en la figura 19.

Bucle para cada tupla (s-a-tipo de recompensa) de la LEAM

Si tipo de recompensa = "recompensa":

Si $a = arg\max_a Q(s,a) \in A(s)$

Break

Sino

$a = arg\max_a Q(s,a) \in A(s)$ + 100

Si tipo de recompensa = "penalidad"

Si $a = arg\min_a Q(s,a) \in A(s)$

Break

Sino

$a = arg\min_a \in A(s)$ - 100

Agregar tupla (s-a-tipo de recompensa) a la LEM

Figura 17. Pseudocódigo del análisis de LEAM.

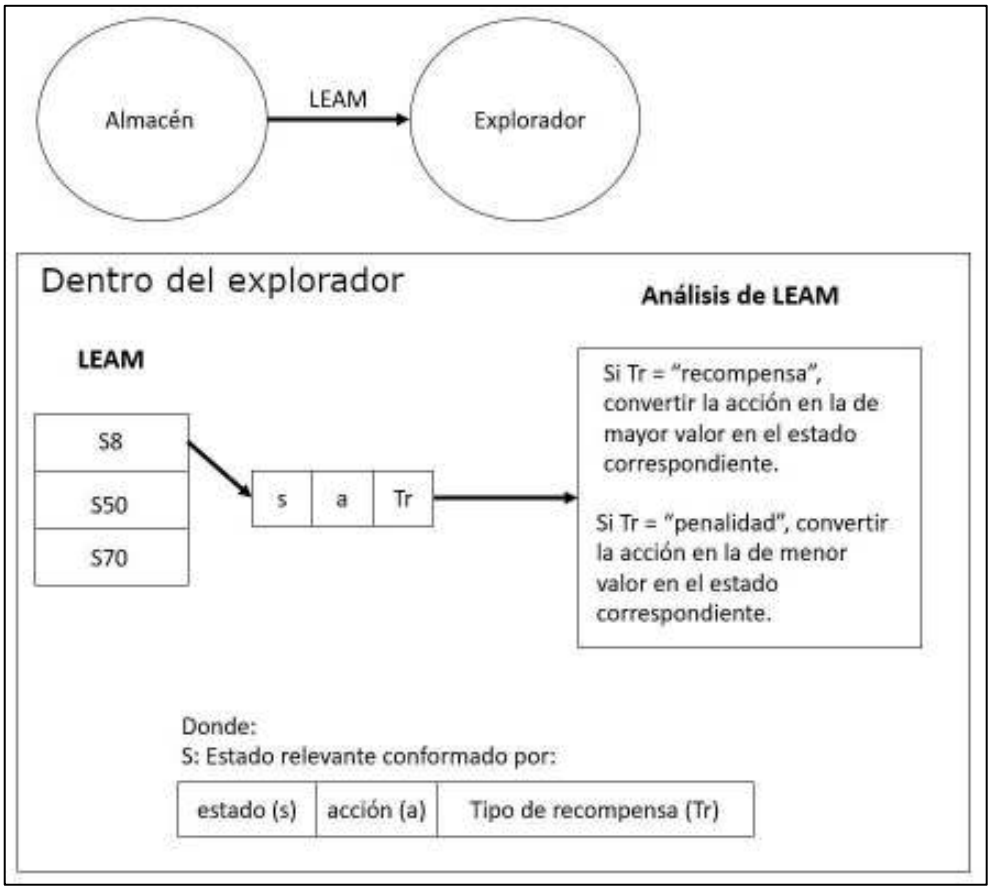


Figura 18. Representación gráfica del análisis de LEAM.

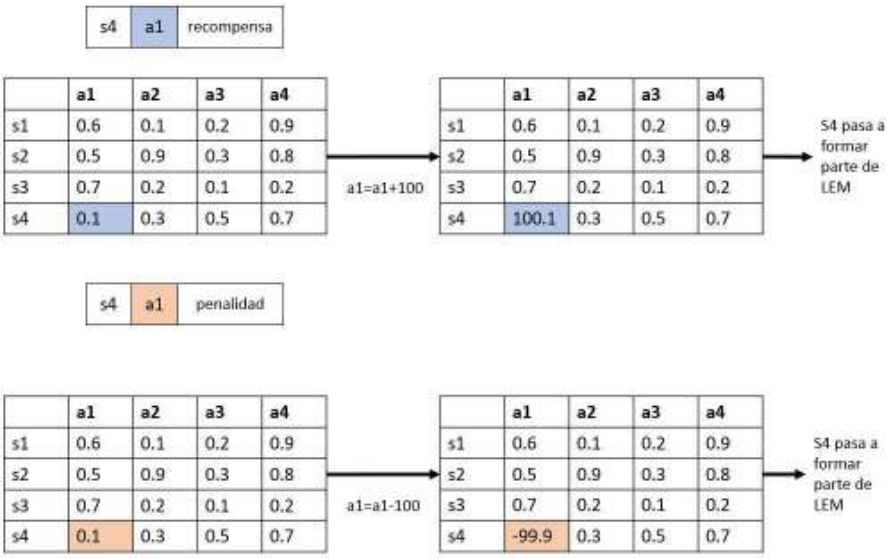


Figura 19. Ejemplos de cambio de valores en base a la recompensa del estado relevante.

Otro cambio que se implementó fue la utilización limitada de $\varepsilon - greedy$. Esta limitación involucra que, si el estado actual se encuentra en LEM, el agente no utiliza $\varepsilon - greedy$, sino que directamente toma la acción de mayor valor. Este uso limitado de $\varepsilon - greedy$, permitió que, si el agente llegó a un estado ya optimizado, directamente explota la mejor acción. De esa manera el agente evitó la posibilidad de explorar un estado que podría llevarlo a obtener una penalidad. Este proceso se muestra de manera gráfica en la figura 20.

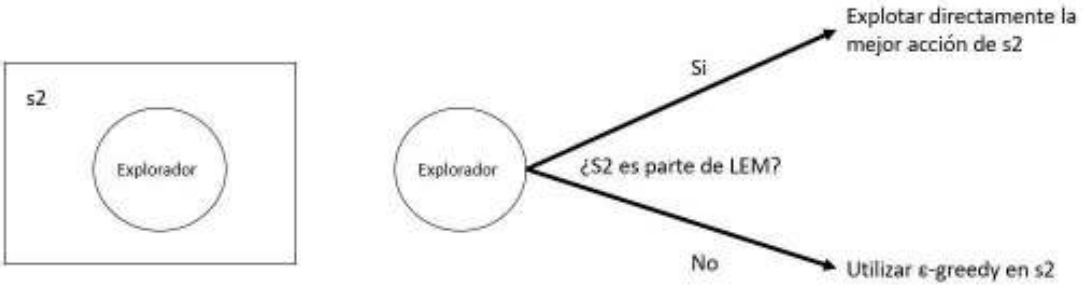


Figura 20. Restricción del uso de $\varepsilon - greedy$

En el ambiente “Mesa”, se implementó un solo envío de información (EI), el cual se produce cuando el agente sobrepasó el borde. Las tuplas enviadas tuvieron como tercer componente la palabra “penalidad”. Esto permitió que tras el análisis de LEAM, la acción que llevó al agente a sobrepasar el borde sea descartada.

En cambio, en el ambiente “Carrera de obstáculos” se implementaron los dos tipos de recompensas, “recompensa” y “penalidad”. Por lo tanto, el EI se produjo cuando el agente llegó a la meta o cuando chocó con un obstáculo. Esto implica que al analizar LEAM, se realizaron los procesos de maximizar un valor y de minimizar un valor.

En el ambiente “carrera de obstáculos” el objetivo no se limitó a que el agente llegue a la meta, sino que también lo logre en el menor número de pasos. Para lograrlo se introdujo una nueva lista y un procedimiento extra. La nueva lista se denominó la “lista de estados de ruta óptima” (LERO). Si el agente llegó a la meta en menos pasos LERO tenía menos elementos.

La figura 21 muestra que una vez que se analizó LEAM y se actualizó LEM, se realizó una comparación entre LEM y LERO, en donde los estados que fueron encontrados en LEM y no en LERO se removían de LEM.

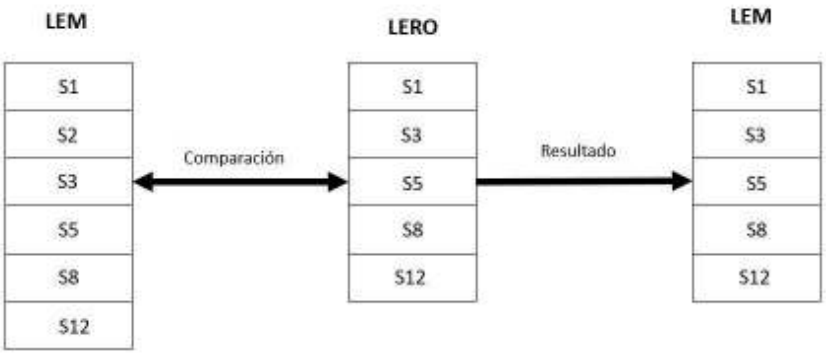


Figura 21. Eliminación de elementos de LEM en base a los elementos de la ruta óptima.

Se estableció también que cada estado atravesado por el agente se almacenara en tuplas que contienen además la acción tomada y “recompensa” como tipo de recompensa. Estas tuplas se almacenaron en una lista llamada “RutaAgente”. Si la meta era alcanzada por el agente, el tamaño de su ruta era comparado con el tamaño de la ruta óptima. Si esta nueva ruta encontrada por el agente era de menor tamaño que la de ruta óptima, se establecía como la nueva ruta óptima. Una vez que se estableció una nueva ruta óptima, las tuplas que se encuentran en esta ruta pasaron a formar parte de LEM y también se enviaron a los otros agentes. Este proceso limitó a que LEM esté formada únicamente por los estados que llevan al agente a la meta en el menor número de pasos. Este proceso se puede observar de manera gráfica en la figura 22 y en pseudocódigo en la figura 23.

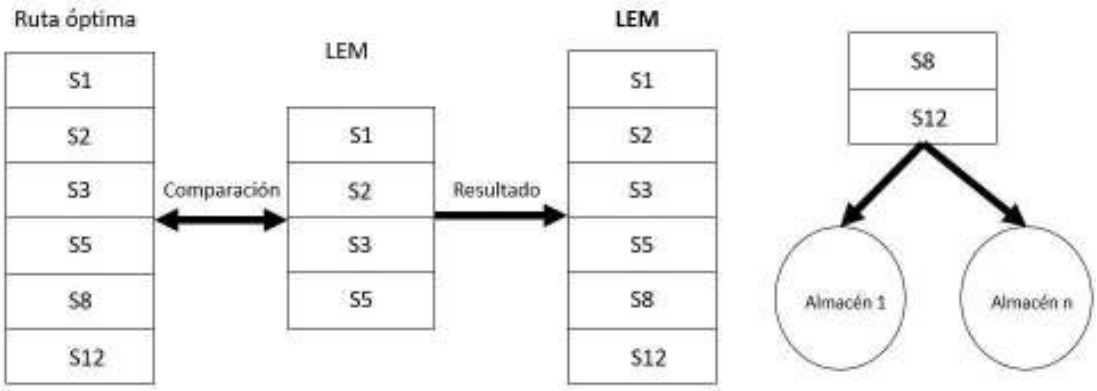


Figura 22. Adición de elementos a LEM en base a los elementos de la ruta óptima.

```
TamañoRutaÓptima = Tamaño del ambiente
Por cada s que el agente atraviesa:
        Agregar tupla [s, a, "recompensa"] a RutaAgente
Si el agente llega a la meta:
        Si Tamaño de RutaAgente < TamañoRutaÓptima
                RutaOptima = RutaAgente
                TamañoRutaÓptima = tamaño de RutaAgente
                        Para cada tupla en RutaOptima:
                                Si tupla no está en LEM
                                        Agregar tupla a LEM
                                        Enviar tupla a los demás agentes
```

Figura 23. Pseudocódigo de la adición de elementos a LEM en base a los elementos de la ruta óptima.

Para cuantificar la diferencia que existe entre el número de interacciones obtenido al utilizar la arquitectura propuesta y el obtenido al no utilizarla, se hizo uso de la fórmula del porcentaje de decremento.

Debido a que existe más de un agente autónomo y al menos uno de ellos utiliza aprendizaje por refuerzo, se trata de un problema de aprendizaje por refuerzo multi agente [22].

En cuanto a la transferencia de aprendizaje, el presente trabajo forma parte de la categoría de transferencia inter-agente. Esto debido a que los agentes trataron de combinar la información transferida con su propia experiencia. Dentro de esta categoría el trabajo planteado se ubica dentro de un grupo más limitado que se caracteriza por aconsejar acciones entre agentes. La tabla 3 ubica al presente trabajo junto a otras investigaciones que utilizan el mismo principio de aconsejar acciones dentro de la transferencia de aprendizaje. La descripción de cooperativo se obtuvo a partir de que los agentes tienen un objetivo en común y se asume que dichos agentes nunca perjudicarán el desempeño de otro. En cuanto a la selección de la tarea de origen y el mapeo de autonomía, el presente trabajo se ubica dentro de la característica de implícito debido a que la reutilización del aprendizaje ocurre únicamente dentro del mismo dominio.

Existen diferencias entre el trabajo propuesto con los trabajos de la tabla 3. El presente trabajo no recibe retroalimentación de un humano como en el trabajo de Griffit [24]. El presente trabajo tampoco hace uso del marco de trabajo del profesor-estudiante [25], [26], [27] y [28]. El presente trabajo no incluye un método en el que los agentes aprendan a cómo o cuándo brindar su consejo, como sucede en los trabajos de Fachantidis et al. (2018) y Omidshafiei et al (2018).

Referencia	Algoritmo de aprendizaje	Selección de la tarea de origen	Mapeo de autonomía
Griffith et al., 2013 [23]	Voraz, adversario, cooperativo	Implícito	Implícito
Torrey & Taylor, 2013 [24]	Cooperativo	Implícito	Implícito
Zhan, Bou-Ammar, & Taylor [25]	Cooperativo	Implícito	Implícito
Amir et al., 2016 [26]	Cooperativo	Implícito	Implícito
Silva et al, 2017 [27]	Voraz, basado en equilibrio, colaborativo	Implícito	Implícito
Fachantidis, Taylor, & Vlahavas, 2018 [28]	Cooperativo	Implícito	Implícito
Omidshafiei et al., 2018 [29]	Cooperativo	Implícito	Implícito
Presente trabajo	Cooperativo	Implícito	Implícito

Tabla 3. Comparación entre el trabajo propuesto y trabajos que utilizan el mismo principio de aconsejar acciones

4. Resultados y Discusión

4.1 Resultados del ambiente "Mesa"

Resultados sin comunicación

Los resultados del agente sin comunicación al momento de aprender una política óptima muestran que el agente alcanzó un rango de recompensa máxima después del episodio número 30000, como se observa en la figura 24 (a). Se trata de un rango de recompensa máxima debido a la ecuación 13. En cuanto a la incidencia de los errores, el agente dejó de cometer errores cerca del episodio 40000, como se observa en la figura 24 (b).

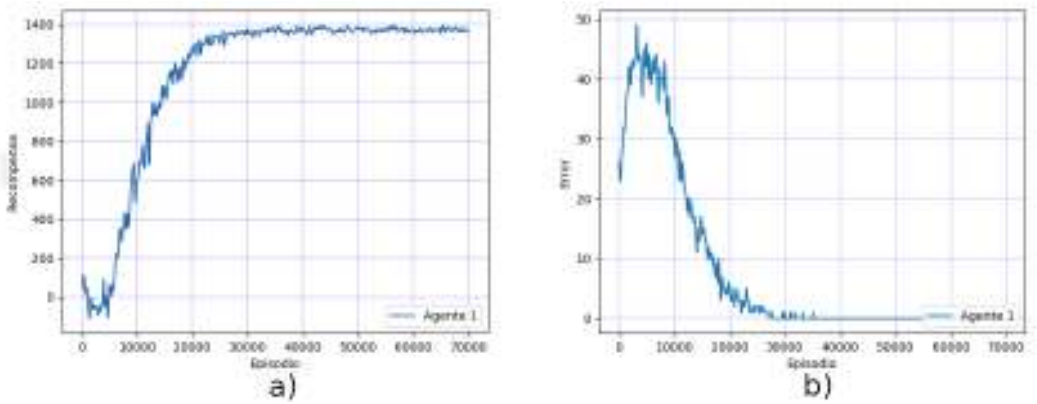


Figura 24. a) Recompensa en función del número de episodios b) Incidencia de errores en función del número de episodios. Para 1 agente sin comunicación durante la fase de entrenamiento en el ambiente "Mesa".

Al momento de poner a prueba la política aprendida por el agente, la recompensa osciló entre un valor de 1470 y 1230, como se observa en la figura 25 (a), también como consecuencia de la ecuación 13. En cambio, la figura 25 (b) muestra que la política aprendida era óptima ya que el agente no cometió errores.

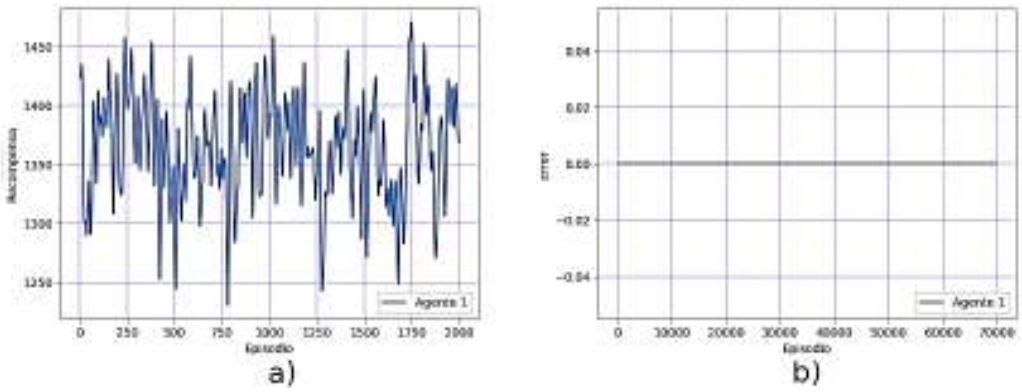


Figura 25. a) Recompensa en función del número de episodios b) Incidencia de errores en función del número de episodios. Para 1 agente sin comunicación durante la fase de ejecución en el ambiente "Mesa".

Resultados utilizando la arquitectura multi agente

La figura 26 muestra los resultados para 1 (a), 3(b), 5(c) y 10 (d) parejas de agentes. Para los 4 casos, los agentes llegaron a un margen de recompensa máxima posible, alrededor de los 17500 episodios.

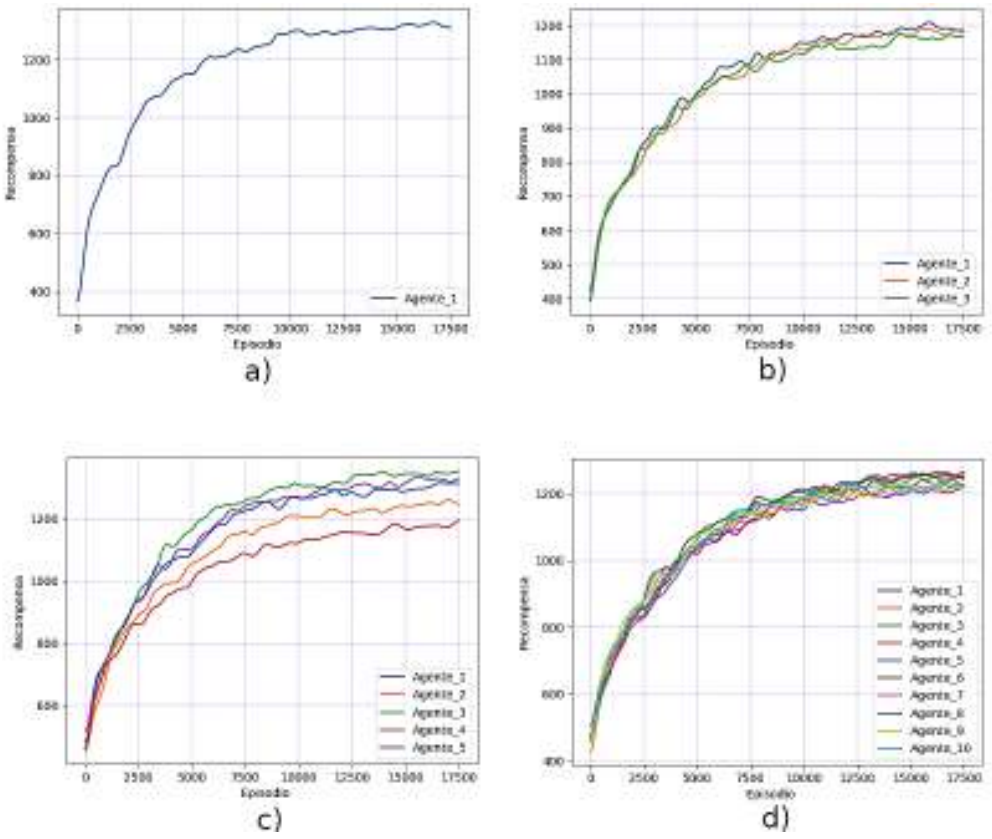


Figura 26. Recompensa en función del número de episodios durante la fase de entrenamiento en el ambiente "Mesa" para: a) 1 pareja de agentes, b) 3 parejas de agentes, c) 5 parejas de agentes, d) 10 parejas de agentes.

La figura 27 muestra la incidencia de errores cada 100 episodios. La figura 27 (a) muestra que para 1 pareja de agentes el número de episodios en el que desaparecieron los errores no supera los 4300. En cambio, las figuras 27 (b), 27 (c) y 27 (d), muestran la incidencia de errores para 3, 5 y 10 parejas de agentes, respectivamente. Se puede apreciar que la incidencia de errores llegó a 0 antes de los 1600 episodios.

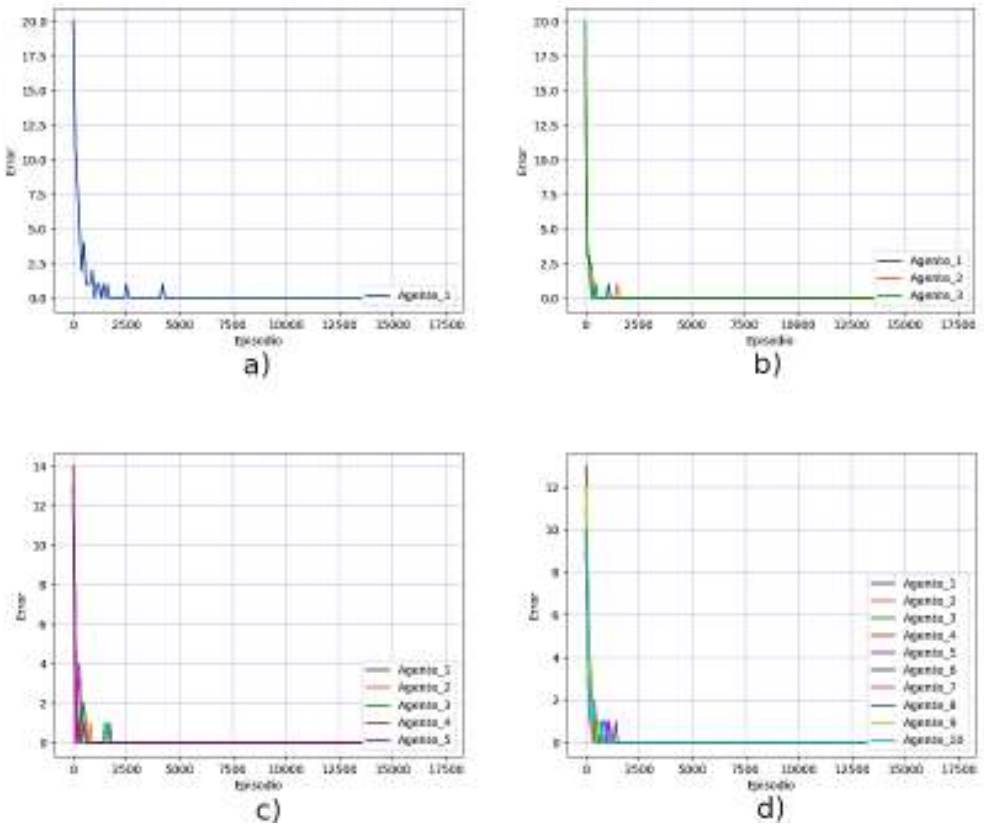


Figura 27. Incidencia de errores en función del número de episodios durante la fase de entrenamiento en el ambiente "Mesa" para: a)1 pareja de agentes, b) 3 parejas de agentes, c) 5 parejas de agentes, d) 10 parejas de agentes.

La figura 28 (a) muestra que al momento de poner a prueba las políticas aprendidas por 10 parejas de agentes, la recompensa máxima obtenida osciló entre 1200 y 950 aproximadamente, también como consecuencia de la ecuación 13. Por otro lado, la figura 28 (b) muestra que los agentes no cometieron errores, por lo tanto, se trataron de políticas óptimas para el caso del ambiente "Mesa".

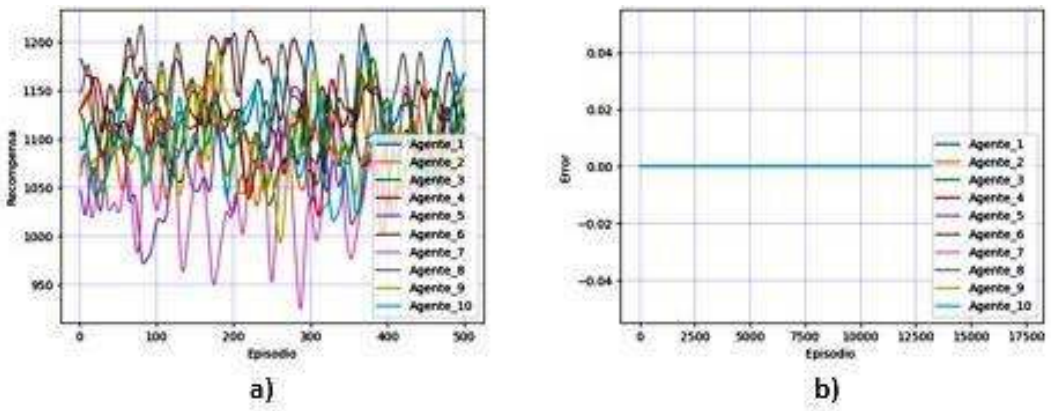


Figura 28. Recompensa en función del número de episodios b) Incidencia de errores en función del número de episodios. Para 10 parejas de agentes sin comunicación durante la fase de ejecución en el ambiente "Mesa".

4.2 Resultados del ambiente "Carrera de obstáculos"

Resultados sin comunicación

La figura 29 (a) muestra que un agente sin comunicación obtuvo la recompensa máxima alrededor de los 51222 episodios. Mientras que, la figura 29 (b) muestra que pasados los 51222 episodios la incidencia de errores se redujo a 0.

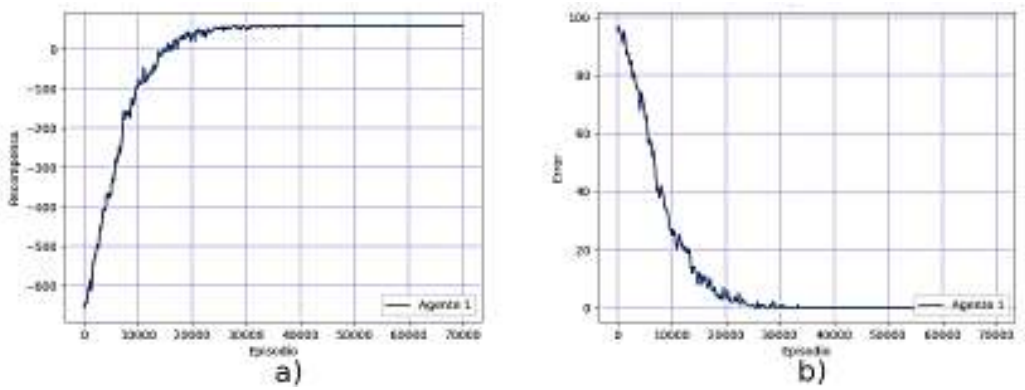


Figura 29. Recompensa en función del número de episodios b) Incidencia de errores en función del número de episodios. Para 1 agente sin comunicación durante la fase de entrenamiento en el ambiente "Carrera de obstáculos".

Resultados utilizando la arquitectura multi agente.

Para todos los gráficos de la figura 30 se muestra una curva suave del incremento de la recompensa en función de los episodios. En la figura 30 a) el número de episodios que le tomó a 1 pareja de agentes alcanzar una recompensa máxima apenas no supera los 125, específicamente el valor obtenido fue de 108. La figura 30 (b) muestra que, en el caso de 3 parejas de agentes, uno de ellos alcanzó la recompensa máxima en alrededor de 50 episodios, mientras que los agentes restantes alcanzaron la recompensa máxima en menos de 100 episodios. La figura 30 (c) muestra los resultados de utilizar 5 parejas de agentes, en este caso también es visible que uno de los agentes tardó alrededor de 50 episodios en

alcanzar la recompensa máxima. En la figura 30 (d), en donde intervinieron 10 parejas de agentes, se observa un ascenso suave de la recompensa, la cual se mantuvo en un valor máximo a partir del episodio 74. Tanto para el caso de 5 y 10 parejas de agentes, el número de episodios en el que todo el sistema alcanzó la recompensa máxima está alrededor de los 125 episodios.

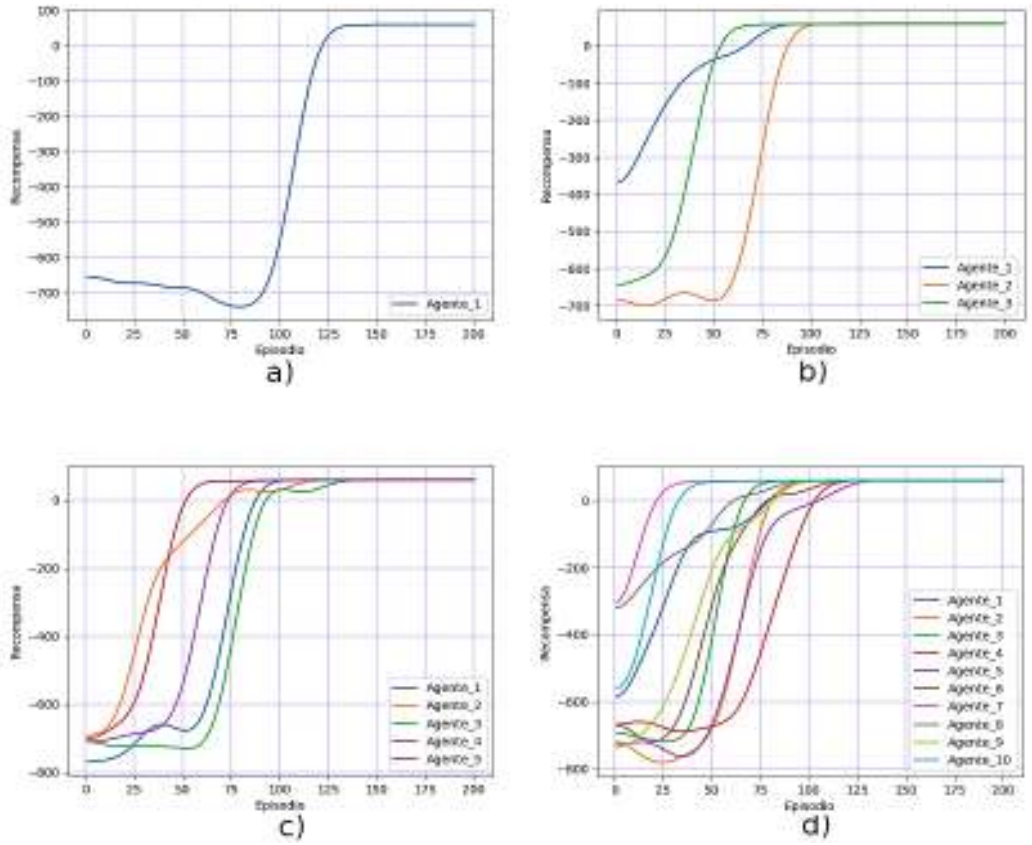


Figura 30. Recompensa en función del número de episodios durante la fase de entrenamiento en el ambiente "Carrera de obstáculos" para: a) 1 pareja de agentes, b) 3 parejas de agentes, c) 5 parejas de agentes, d) 10 parejas de agentes.

En cuanto a la incidencia de errores, la figura 31 muestra la incidencia de errores cada 100 episodios. Se puede apreciar en la figura 31 (a) que, para 1 pareja de agentes, los errores desaparecieron antes de 110 episodios, específicamente los valores desaparecieron en el episodio 107. Para el caso de 3 parejas de agentes, el agente que más tardó en llegar a 0 errores no superó los 75 episodios, como se observa en la figura 31 (b). Mientras que el agente que menos tardó en reducir sus errores tardó 40 episodios. Para 5 parejas de agentes, como se aprecia en la figura 31 (c), se observa que el agente que menos tardó en reducir a 0 la incidencia de errores no superó los 40 episodios. Para 10 parejas de agentes, como se observa en la figura 31 (d), uno de los "exploradores" no superó los 10 episodios en reducir sus errores a 0. Tanto para el caso de 5 y 10 parejas de agentes los errores en todo el sistema desaparecieron antes de 120 episodios.

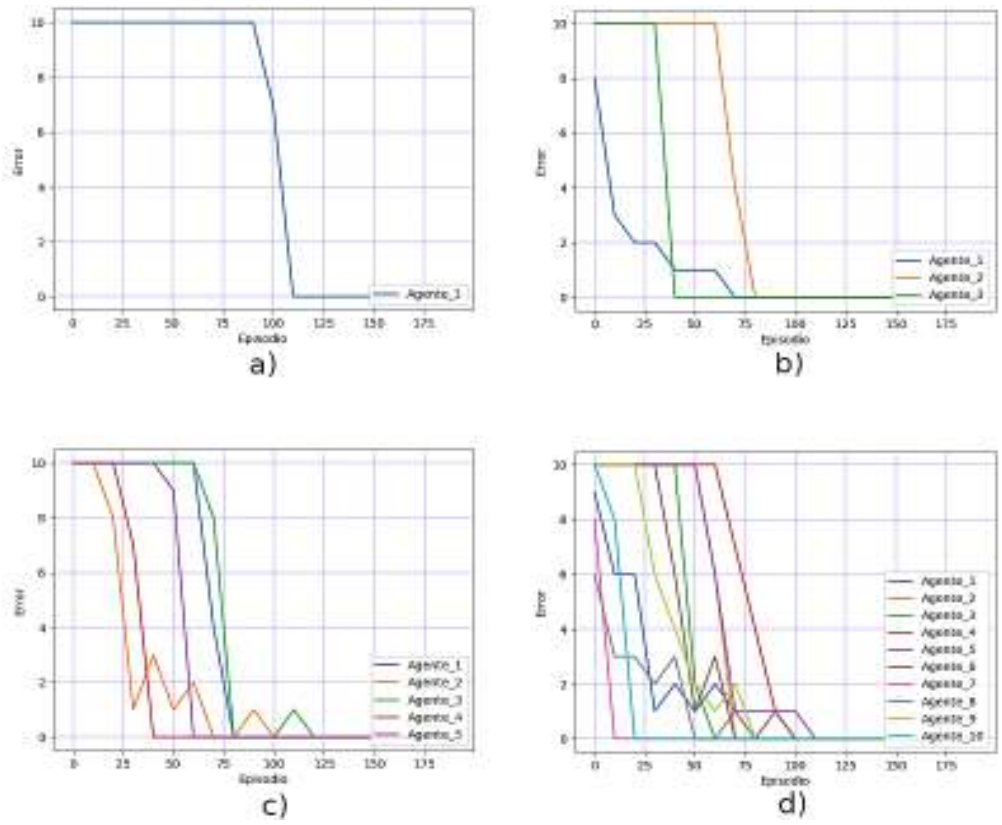


Figura 31. Incidencia de errores en función del número de episodios durante la fase de entrenamiento en el ambiente "Carrera de obstáculos" para: a)1 pareja de agentes, b) 3 parejas de agentes, c) 5 parejas de agentes, d) 10 parejas de agentes.

La figura 32 (a) muestra que los agentes obtuvieron la recompensa máxima. Por otro lado, la figura 32 (b) muestra que los agentes no cometieron errores.

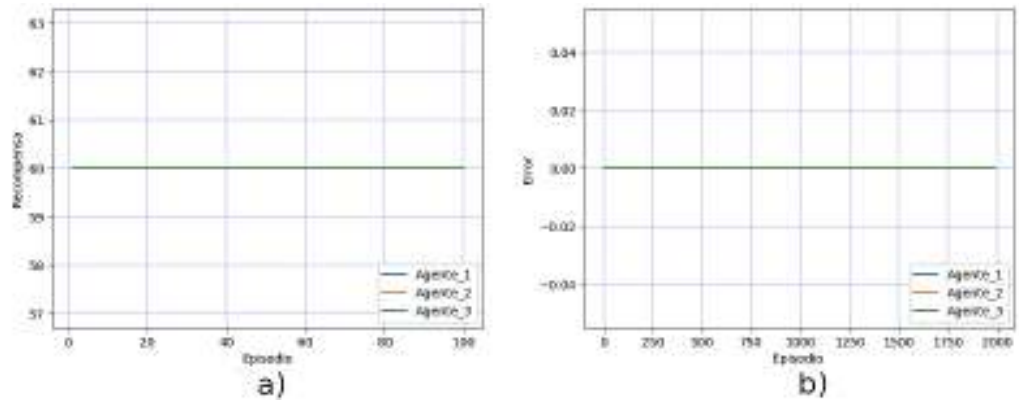


Figura 32. a) Recompensa en función del número de episodios b) Incidencia de errores en función del número de episodios. Para 3 parejas de agentes sin comunicación durante la fase de ejecución en el ambiente "Carrera de obstáculos".

4.3 Discusión

Los resultados del ambiente "Carrera de obstáculos" muestran que existe un agente dentro del sistema al que le tomó un mayor número de interacciones alcanzar la política óptima. Este

hallazgo guarda una similitud con el concepto de “agente perezoso” [6]. Sin embargo, aunque el agente tarda en llegar a una política óptima, no se considera un problema ya que eventualmente obtuvo el conocimiento necesario de sus similares y logró llegar al objetivo.

Existen varios enfoques de sistemas multi agente en donde existe cooperación entre agentes con el fin de resolver el estancamiento [20]. Dentro de la arquitectura multi agente propuesta, los agentes cooperan mediante el intercambio de información. Esta información estuvo conformada por estados, acciones y recompensas, que tras su análisis (figura 14), modificaron la función Q de los agentes permitiéndoles evitar el estancamiento.

La cooperación entre agentes se observó de mejor manera en el ambiente “Mesa”, en donde los agentes modificaron estados comunicados por sus similares. Este aporte de conocimiento evitó que los agentes se equivoquen o estanquen en estados aún no explorados.

En el ambiente “Mesa” los agentes únicamente descartaron la acción que los llevó a equivocarse en cierto estado. Al descartar la acción y al encontrarse el estado ya en LEM, el agente explotó la mejor acción para ese estado, y dicha acción no siempre fue la misma. Esto provocó que los agentes, a pesar de encontrarse en el mismo estado, se muevan en direcciones diferentes. En contraposición, los agentes en el ambiente “carrera de obstáculos” llegaron a un mismo comportamiento, dado que convergen a una sola ruta óptima.

Los resultados en los ambientes “Mesa” y “Carrera de obstáculos”, muestran que existe un decremento en el número de interacciones necesarias para mejorar el comportamiento de los agentes. Esto se debe a que el uso de múltiples agentes permitió acelerar la exploración del ambiente, un resultado esperado si se toma como referencia el trabajo de Majumdar [30].

. Se puede observar que el mejor resultado en cuanto al número de episodios necesarios para que el agente no cometa errores se obtuvo con 10 parejas de agentes ya que representa un decremento del 98.31%. Para los otros casos con comunicación, el decremento es del 90.55%, 97.7% y 97.4%, para los casos de 1, 3 y 5 parejas de agentes respectivamente. Para todos los casos con comunicación, el número de episodios en los que los agentes alcanzaron un rango máximo de recompensa muestra un decremento del 61.1%.

Se observa que el número de episodios en el que los agentes alcanzan un rango máximo no varía de acuerdo con el número de agentes. Esto se debe a que los agentes aprenden de sus similares a no caer de la mesa, pero de manera individual a maximizar la recompensa. Dado que alcanzar la recompensa máxima involucra que el agente se mantenga en el borde, existe una relación con la ecuación 13. Esta relación se refleja en que a pesar de que el agente no

cae de la mesa, llegar a moverse cerca del borde toma un número mayor de episodios para cada agente, equivalente a aproximadamente 17500.

Sin comunicación		
N° de agentes	Episodios en los que el agente comenzó a obtener una recompensa que oscila en un rango de recompensa máxima.	Episodios en donde el agente dejó de cometer errores.
1	45102	45102
Con comunicación (Uso de la arquitectura propuesta)		
N° de parejas de agentes	Episodios en los que el agente comenzó a obtener una recompensa que oscila en un rango de recompensa máxima.	Episodios en donde el agente dejó de cometer errores.
1	17 500 aprox.	4283
3	17 500 aprox.	1030 *
5	17 500 aprox.	1136 *
10	17 500 aprox.	760 *

Tabla 4. Resumen de resultados en el ambiente "Mesa"

*Promedio entre el valor obtenido por el agente que menos tardó en llegar al objetivo y el agente que más tardó en alcanzar el objetivo (*midrange*)

Las figuras 18 y 21 evidencian que el rango de la recompensa máxima es más alto en un entorno sin comunicación. Esto se debe a un denominado "efecto rebote" que mostraron los agentes el cual, no permite que el agente se mantenga en el borde obteniendo la recompensa máxima que ofrece el ambiente. Este efecto, que se visualiza en el anexo 1, se produjo debido a que el algoritmo del trabajo propuesto solamente hizo que el agente descartara la acción que lo llevó a obtener la penalidad. Esto no aseguró que de ahora en adelante la acción explotada por el agente lo mantenga en el borde. En cambio, en un ambiente sin comunicación, el agente continuó explorando hasta lograr que su mejor acción sea aquella que lo mantenía en el borde de la mesa.

Sin embargo, es importante destacar que a pesar de que los valores máximos de la recompensa son menores, el objetivo del agente era no sobrepasar el borde la mesa, el cual fue alcanzado en aproximadamente la mitad de los episodios.

Por otro lado, para el ambiente "Carrera de obstáculos", los resultados se encuentran en la tabla 5. Al comparar los resultados del modelo sin comunicación con el modelo utilizando la arquitectura para 1 pareja de agentes, el número de episodios para que no existan equivocaciones se redujo un 99.79%. El número de episodios para alcanzar solo la recompensa máxima también se redujo un 99.78%. Sin embargo, para una pareja de agentes,

la ejecución tomó mucho más tiempo, dado que el agente “Explorador”, debió esperar a que el agente “Almacén” finalice con sus procesos para poder iniciar el siguiente episodio. Esto indicó que era necesario utilizar más de una pareja de agentes.

Para el caso de 3 parejas de agentes, el número de episodios para que no existan equivocaciones se redujo un 99.89%. El número de episodios para maximizar la recompensa se redujo un 99.88%. Para 5 parejas de agentes, el número de episodios en los que los agentes alcanzaron únicamente la recompensa máxima se redujo en un 99.83%. Mientras que, el número de episodios en que los errores desaparecen se redujo en 99.85%. Para el caso de 10 parejas de agentes, el número de episodios que le toma al agente alcanzar solo la máxima recompensa logró un decremento del 99.80%. Por otro lado, el número de episodios en que el agente llevó sus errores a 0 se redujo en un 99.88%.

Sin comunicación		
N° de agentes	Episodios en los que el agente obtiene únicamente la recompensa máxima	Episodios en donde el agente dejó de cometer errores.
1	51222	51222
Con comunicación (Uso de la arquitectura propuesta)		
N° de parejas de agentes	Episodios en los que el agente obtiene únicamente la recompensa máxima	Episodios en donde el agente dejó de cometer errores.
1	108	107
3	59*	52 *
5	86 *	75*
10	100 *	58 *

Tabla 5. Resumen de resultados en el ambiente “Carrera de obstáculos”

5. Conclusiones y Recomendaciones

Los resultados de poner a prueba la arquitectura multi agente, indican que el número de episodios para que el agente no cometa errores en el ambiente se reduce en más de un 90%, reduciendo así el número de interacciones agente-ambiente Esto indica que la arquitectura propuesta efectivamente, comunica el conocimiento necesario para que los agentes mejoren su comportamiento.

La arquitectura multi agente propuesta presenta un modelo de trabajo en pares. Para que la arquitectura funcione correctamente, el número de agentes “explorador” y “almacén” debe ser el mismo. Un agente “explorador” sin pareja “almacén” permanece en espera de comunicarse y no procede a explorar el ambiente. En cambio, un agente “almacén” sin pareja “explorador” solamente acumula los conocimientos, pero no le es de utilidad al sistema. En cambio, el

número de parejas de agentes necesario para obtener mejores resultados depende del ambiente.

Los modelos de aprendizaje por refuerzo funcionan como el medio a partir del cual los agentes obtienen su conocimiento. Este conocimiento, reflejado en estados acciones y recompensas, es enviado a través de los agentes y asimilado por ellos con el fin de cambiar su comportamiento.

Tras la integración del aprendizaje por refuerzo con la arquitectura multi agente se evidencia que la modificación del algoritmo Q-*learning*, permite que los agentes mejoren su comportamiento. Específicamente, el análisis de estados relevantes y la limitación del uso de e-greedy, permite que los agentes aprovechen el conocimiento de sus similares y aprendan más rápido sobre el ambiente.

Las tablas 4 y 5 muestran que es mejor utilizar la arquitectura multi agente que utilizar un solo agente. Los resultados del ambiente “mesa” muestran que los mejores resultados se obtienen con 10 parejas de agentes. Esto se debe a que, al ser un número mayor de agentes, se puede explorar más estados de la mesa a la vez. En cambio, en el ambiente “carrera de obstáculos”, los mejores resultados se obtienen con 3 parejas de agentes. Esto indica que no es necesario incrementar el número de agentes ya que todos convergen a la misma ruta.

6. Referencias

[1] M. Wooldridge, An Introduction to Multi Agent Systems, 2nd ed. John Wiley & Sons, 2009.

[2] R. Sutton and A. Barto, Reinforcement learning An Introduction, 2nd ed. The MIT Press, 2018.

[3] L. Squire, Fundamental neuroscience. Amsterdam: Elsevier/Academic Press, 2008.

[4] A. Fusté, J. Amores, D. Ha, J. Jongejan, and A. Pitaru, “Paper Cubes: Evolving 3D characters in Augmented Reality using Recurrent Neural Networks”. 2017.

[5] L. Kraemer and B. Banerjee, "Multi-agent reinforcement learning as a rehearsal for decentralized planning", Neurocomputing, vol. 190, pp. 82-94, 2016.

[6] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls. “Value-decomposition networks for cooperative multi-agent learning”, 2017.

[7] K. S. Hwang, J. L. Lin and H. P. Hsu, "A multi-agent cooperation system based on a Layered Cooperation Model", 2014 IEEE International Conference on System Science and Engineering (ICSSE), 2014.

[8] P. Dolan, M. Hallsworth, D. Halpern, D. King, and I. Vlaev, “MINDSPACE: Influencing behaviour through public policy”, 2010.

[9] N. Jennings, "On agent-based software engineering", Artificial Intelligence, vol. 117, no. 2, pp. 277-296, 2000.

[10] N. Vlassis and R. Brachman, A Concise Introduction to Multiagemnt Systems and Distributed Artificial Intelligence. Oregon, State University: Morgan & Claypool Publishers, 2007.

[11] P. Balaji and D. Srinivasan, "An Introduction to Multi-Agent Systems", Innovations in Multi-Agent Systems and Applications 1, pp. 1-27, 2010.

[12] Y. Achbany, F. Fouss, L. Yen, A. Pirotte, Marco Saerens, “Managing the Exploration/Exploitation Trade-Off in Reinforcement Learning”, 2005.

[13] S. Wang & Y. Jing, “Deep Reinforcement Learning with Surrogate Agent-Environment Interface”, 2017.

[14] E. Zanini, “Markov Decision Process”, 2005.

[15] M. Otterlo, “Markov Decision Processes: Concepts and Algorithms”, 2009.

[16] S. Ivanov and A. D'yakonov, “Modern Deep Reinforcement Learning Algorithms”, 2019.

[17] C. Watkins and P. Dayan, "Q-learning", Machine Learning, vol. 8, no. 3-4, pp. 279-292, 1992.

[18] M. Tokic, "Adaptive ε-Greedy Exploration in Reinforcement Learning Based on Value Differences", KI 2010: Advances in Artificial Intelligence, pp. 203-210, 2010.

[19] D. Kim, et al. “Learning to Schedule Communication in Multi-agent Reinforcement Learning.” ,2019.

[20] J. N. Foerster, M.A. Yannis, Nando de Freitas and S. Whiteson. “Learning to Communicate with Deep Multi-Agent Reinforcement Learning”, 2016.

[21] A. Dorri, S. Kanhere and R. Jurdak, "Multi-Agent Systems: A Survey", IEEE Access, vol. 6, pp. 28573-28593, 2018.

[22] F. Silva and A. Costa, "A Survey on Transfer Learning for Multiagent Reinforcement Learning Systems", Journal of Artificial Intelligence Research, vol. 64, pp. 645-703, 2019.

[23] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. L. Thomaz “PolicyShaping: Integrating Human Feedback with Reinforcement Learning”, Advances in Neural Information Processing Systems (NIPS), pp. 2625–2633, 2013.

[24] L. Torrey and M. E. Taylor, “Teaching on a Budget: Agents Advising Agents in Reinforcement Learning”, Proceedings of 12th the International Conference on Au-tonomous Agents and Multi Agent Systems (AAMAS), pp. 1053–1060, 2013.

[25] Y. Zhan, H. Bou-Ammar, and M. E. Taylor, “Theoretically-Grounded Policy Advice from Multiple Teachers in Reinforcement Learning Settings with Applications to Neg-ative Transfer”, Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI), pp. 2315–2321, 2016.

[26] O. Amir, E. Kamar, A. Kolobov, and B. Grosz, “Interactive Teaching Strategies for Agent Training”, Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI), pp. 804–811, 2016.

[27] F. Silva, R. Glatt, and A. Costa, “Simultaneously Learning and Advising in Multiagent Reinforcement Learning”, Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 1100–1108, 2017

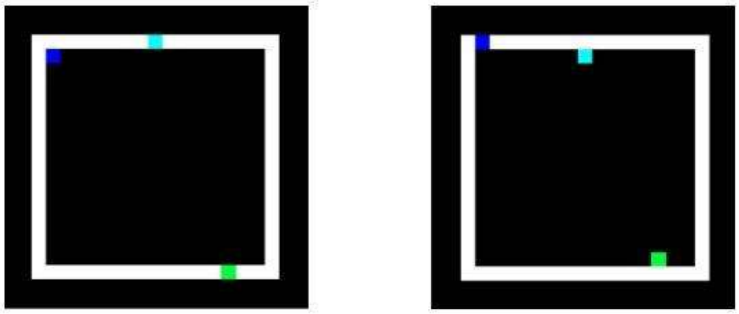
[28] A. Fachantidis, M. Taylor, & I. Vlahavas, “Learning to Teach Reinforcement Learning Agents”, Machine Learning and Knowledge Extraction, 2018.

[29] S. Omidshafiei, et al. “Learning to Teach in Cooperative Multiagent Reinforcement Learning”, Workshop on Lifelong Learning: A Reinforcement Learning Approach, 2018.

[30] A. Majumdar, P. Benavidez and M. Jamshidi, "Multi-Agent Exploration for Faster and Reliable Deep Q-Learning Convergence in Reinforcement Learning," 2018 World Automation Congress (WAC), Stevenson, WA, 2018, pp. 1-6, 2018.

7. Anexos

7.1 Anexo 1. Visualización del “efecto rebote”



a) b)

El efecto consiste en que cuando el agente llega al borde de la mesa, retrocede un espacio y regresa al borde. Los agentes repiten los movimientos vistos en las figuras 1 (a) y (b) hasta agotar sus movimientos.