

Gian Nestor Cuello Nicholson

2017

Resumen

La tesis trata sobre resolver un problema de formación de múltiples agentes para recorrer un área discreta, este problema esta considerado en dos dimensiones y tres dimensiones. El trabajo desarrollado propone el uso de una técnica de la Inteligencia Artificial para encontrar soluciones optimas de ambos problemas, mediante el uso de aprendizaje por refuerzo, los cuales son algoritmos para encontrar una póliza optima basado en una mecánica de acción recompensa, un método que ha sido aplicado exitosamente en diversos problemas. El problema de formación de multi-agentes es donde un grupo de agentes tienen que encontrar una ruta óptima para llegar a una posición ideal y entrar en una formación. Aquí se tienen ciertas recompensas dadas para poder adquirir una póliza ideal para la formación. Por otro lado el problema de recorrer un área se implemento una técnica acción-recompensa similar como el primer problema, obteniendo resultados satisfactorios para ambos. Finalmente los aprendizajes obtenidos de ambos problemas son implementados en simulaciones para ver el correcto funcionamiento de lo aprendido.

Índice general

Resumen	I
Contenidos	III
Lista de figuras	V
Agradecimientos	VI
Nomenclatura	VI
1. Introducción	1
1.1. Antecedentes	5
1.2. Organización de la tesis	8
2. Marco teórico	9
2.1. Aprendizaje Automático	9
2.2. Aprendizaje por Refuerzo	10
2.2.1. Procesos de Decisión Markov (MDP)	11
2.2.2. Elementos del aprendizaje por refuerzo	12
2.2.3. Metas y recompensas	13
2.2.4. Estrategias de exploración-explotación	14
2.2.5. Q-Learning	15
2.3. Enjambres Inteligentes	17
2.3.1. Optimización de enjambre de partículas (PSO):	19
3. Metodología	21
3.1. Problema de Formación	21
3.2. Cubrir área discreta	24
3.3. Agentes en mundo de 3-Dimensiones	27
3.4. Simulaciones	30
3.4.1. Funcionamiento general de las simulaciones	31
3.5. Resumen	32

4. Resultados	34
4.1. Escenarios experimentales en dos dimensiones	34
4.1.1. Experimento para barrido de espacio	34
4.1.2. Experimento para formación de agentes	41
4.1.3. Animación en dos Dimesiones	47
4.2. Escenarios experimentales en tres dimensiones	50
4.2.1. Experimento para barrido de espacio	50
4.2.2. Animación en tres Dimesiones	53
4.2.3. Resumen	53
5. Conclusiones	54

Índice de figuras

1.1.	Diagrama de Aprendizaje por Refuerzo	2
1.2.	Diagrama de los sub problemas de la navegación	3
2.1.	Diagrama de aprendizaje por refuerzo	13
2.2.	Diagrama de apoyo Q-Learning	16
3.1.	Diagrama del mundo discreto, donde G es la posición meta y cada agente tiene una posición inicial.	22
3.2.	Diagrama del mundo discreto, donde G es la posición meta y cada agente tiene una posición inicial.	23
3.3.	Diagrama del mundo discreto, donde G es la posición meta y cada agente tiene una posición inicial.	25
3.4.	Diagrama del mundo discreto, donde el agente global se mueve en las cuadrículas, del mundo global (3×3), y cada cuadrícula local es 3×3	27
3.5.	Representación de un espacio en mundo tri-dimensional. Esta compuesto por un grid de $n \times m$ en los tres niveles y solo representa una celda del mundo.	28
3.6.	Representación de los movimientos de un multi-agente en un espacio del mundo en tres dimensiones.	29
3.7.	Figura de un mundo de cuadrícula de 3×3	30
3.8.	Figura de un mundo de cuadrícula en 3 dimensiones con 3 representaciones de agentes	31
3.9.	Diagrama de flujo de simulación.	32
4.1.	Gráficas comparativas de diferencia en Q por episodios con 100 pasos como máximo y 2000 episodios.	35
4.2.	Ilustración que muestra el punto inicial del mundo con índice 0 y finaliza en el índice 6.	37
4.3.	Gráficas comparativas de diferencia en Q por episodios con 100 pasos como máximo y 2000,5000, 10000 y 100000 episodios.	38

4.4. Ilustración que muestra el punto inicial del mundo con índice 0 y finaliza en el índice 6.	40
4.5. Metas de las diferentes direcciones para la formación de 3 agentes en triángulo	41
4.6. Gráficas comparativas de diferencia en Q por episodios con 200 pasos como máximo y 2000 a 10000 episodios.	42
4.7. Formación de 4 agentes en cruz.	43
4.8. Gráficas comparativas de diferencia en Q por episodios con 200 pasos como máximo y 2000 a 10000 episodios.	44
4.9. Metas de las diferentes direcciones para la formación de 4 agentes en triángulo	45
4.10. Gráficas comparativas de diferencia en Q por episodios con 200 pasos como máximo y 2000 a 10000 episodios.	46
4.11. Simulación en 2 dimensiones utilizando el conocimiento de barrido y de formación en triángulo de 3 agentes	47
4.12. Simulación en 2 dimensiones utilizando el conocimiento de barrido y de formación en triángulo de 4 agentes en cruz	48
4.13. Simulación en 2 dimensiones utilizando el conocimiento de barrido y de formación en triángulo de 4 agentes en triángulo	49
4.14. Gráficas comparativas de diferencia en Q por episodios con 100 pasos como máximo y 2000 episodios.	51
4.15. Simulación en 2 dimensiones utilizando el conocimiento de barrido y de formación en triángulo de 4 agentes en triángulo	53

Agradecimientos

Capítulo 1

Introducción

Robots móviles tienen diversas aplicaciones abarcando áreas como el entretenimiento, juguetes, en el campo militar, también utilizados para misiones de rescate sin intervención humana, y utilizados para exploración de otros planetas. Por esto los robots móviles son importantes, por esta habilidad de poder desplazarse.

En la actualidad existen diferentes tipos de robots pero una característica principal que los distingue es su locomoción. La locomoción de un robot móvil puede ser terrestre con llantas, humanoides o con múltiples patas, orugas o cadenas, acuático, y aéreo. Estos robots pueden ser tele-operados o autónomos. Los vehículos autónomos tiene la capacidad de realizar tareas en ambientes dinámicos o fijos sin intervención humana. Para decir que un robot es autónomo, sea aéreo, terrestre o acuático, tiene que cumplir con lo siguiente:

- **Percepción:** Obtener e utilizar el conocimiento del ambiente y de si mismo. Esto se logra sea tomando medidas con sensores y extrayendo lo necesario para utilizar en tareas a futuro.
- **Inteligencia:** Tiene que operar por un periodo de tiempo sin intervención humana. Esto se logra asociando un aprendizaje con capacidades de realizar inferencias, para que el vehículo se adapte al ambiente.
- **Acción:** Movilizarse de un punto A - punto B. El vehículo debe utilizar el conocimiento adquirido para desplazarse en un ambiente dinámico sin involucrar humanos.

El campo de desarrollo para un sistema autónomo a logrado grandes avances para obtener diferentes grados de autonomía para los robots. La autonomía esta ligada al problema de navegación, y para llegar a eso se necesita una constante interacción de las características mencionadas, percepción, inteligencia y acción.

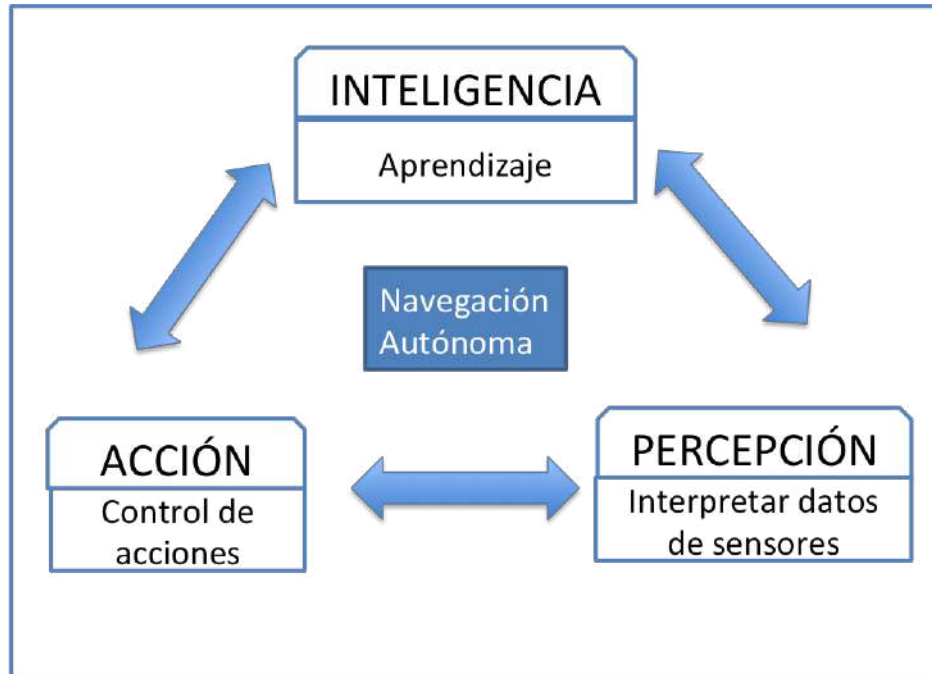


Figura 1.1: Diagrama de Aprendizaje por Refuerzo

A continuación se tiene una ilustración del proceso de interacción de las tres características que tiene que cumplir un robot autónomo:

La navegación es la habilidad que caracteriza a un robot móvil pero al mismo tiempo es uno de los problemas más desafiantes de la robótica móvil. El problema de la navegación puede definirse como: “Dado un punto de partida A alcanzar el (los) punto(s) de destino B (B1, B2, ...) utilizando el conocimiento y la información sensorial recibida del ambiente”. Debido a que un robot no posee los sentidos naturales, que si tiene el ser humano, un robot no puede explorar un entorno desconocido a menos que se le proporcionen fuentes de detección que le permitan obtener información del ambiente. Existe muchos sensores (sonares, odómetros, láser, sistemas de posicionamiento global, cámaras, etc.) que pueden utilizarse para hacer a un robot capaz de detectar una amplia gama de entornos.

La navegación de robots móviles es un tema amplio que cubre grandes aspectos de diferentes tecnologías y aplicaciones. Uno de los aspectos importante a tener en cuenta es la escala física de los requerimientos de navegación, la cual puede medirse por la precisión con la que el robot necesita navegar (esta es la resolución de la navegación). Estos requisitos varían mucho con la aplicación, sin embargo una aproximación de primer orden puede tomarse de las dimensiones del dispositivo mismo. Cualquier dispositivo autónomo debe ser capaz de determinar su posición a una resolución dentro de al menos sus propias dimensiones, con el fin de ser capaz de navegar e interactuar con su entorno correctamente. Una forma común de categorizar esta escala de requerimientos es la siguiente [6]:

- Navegación local.- Es la capacidad de determinar la posición respecto a los objetos (estacionarios o móviles) en el entorno e interactuar con ellos correctamente. Trata con la navegación en la escala de unos pocos metros, donde el principal problema es evitar los obstáculos.
- Navegación global.- Es la capacidad de determinar su posición en términos absolutos o referenciados al mapa y de moverse a un punto de destino deseado. Trata con la navegación en una escala grande en la cual el robot no puede observar el estado meta desde su posición inicial.

Otro aspecto importante a tener en cuenta en la navegación es que el mapa del medio ambiente es una necesidad básica de un robot para realizar servicios, como por ejemplo agarrar un objeto y llevarlo de una

habitación a otra. Para realizar este tipo de servicios, el robot no sólo debe saber sobre el medio ambiente,

sino que mientras esta en movimiento también debe ser consciente de su propia ubicación en ese entorno [30]. Adicionalmente, la representación adecuada del robot mismo en el medio ambiente también juega un papel vital para resolver muchos problemas relacionados con la navegación autónoma.

De forma general el problema de la navegación implica resolver los sub-problemas que se muestran en la figura 1.2, pero existen diferentes enfoques y soluciones para cada sub-problema.

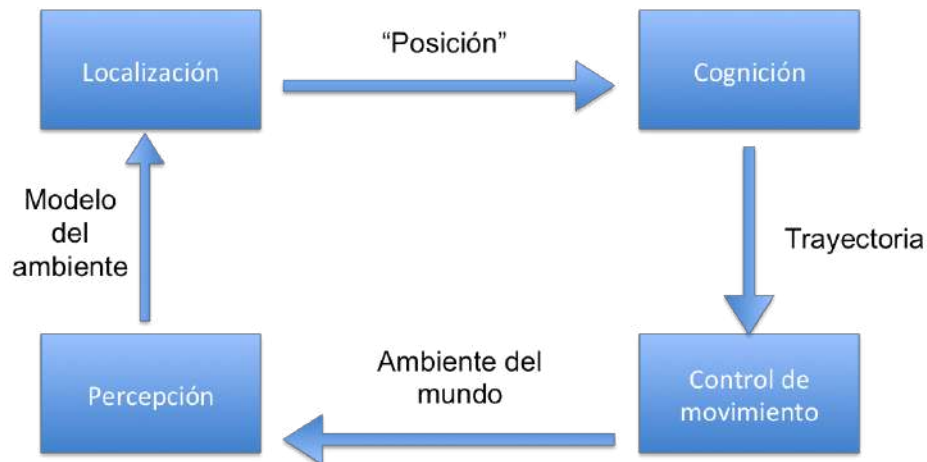


Figura 1.2: Diagrama de los sub problemas de la navegación

Con los avances recientes en robótica se ha popularizado utilizar vehículos aéreos no tripulados (por sus siglas en inglés, VANTS) o drones. Estos vehículos han facilitado diversas tareas que un robot terrestre no puede realizar. Los drones en el campo militar son parte vital de las operaciones para mantener bajo

vigilancia zonas militares, fronteras, seguimiento de personas en campos de batalla, incluyendo la guerra contra el terrorismo. Con el éxito que se ha dado en el campo militar muchos países han optado por implementar el dron como un arma de guerra.

Después de la adopción en el campo militar, la industria inicio a fabricar estos vehículos a bajos costos y por consecuente ahora cualquier persona puede comprar un VANT. En la actualidad la industria de fabricación de drones esta valuada en 10 mil millones de dólares y sigue en crecimiento. Con esta gran demanda de drones sus aplicaciones han variado mucho.

Una aplicación popular de los drones es en campo agrícola, utilizados para escanear terrenos de sembradillos. Esto es útil por que el campesino puede saber si el sistema de riego esta funcionando, a su vez puede ver como sus plantas van creciendo día a día y saber si alguna planta esta enferma. Este robot ayuda mucho en tiempos de desplazamiento que la persona ya no necesita realizar.

No alejándose del campo, otra aplicación que se ha dado para VANTs es para apagar incendios, especialmente bosques en llamas. Aquí, el dron no solamente esta vigilando la zona pero esta habilitado para combatir el fuego. Sin duda alguna esta tecnología ayudado mucho a los cuerpos de bomberos.

Entendido que la industria de drones esta en crecimiento y sus costos son relativamente bajos, ahora podemos pensar en la posibilidad de tener más de un dron en funcionamiento para cumplir con la tarea. Pero si controlar un vehículo ya involucra su conjunto de problemas ahora tener un conjunto de estos se vuelve un problema sumamente complejo. La idea de poder tener un conjunto autónomo de drones se extiende de la forma en la cual la naturaleza funciona. En la naturaleza siempre se ve como los pájaros están en una formación, los pescados también siguen una forma de viajar en conjunto, un enjambre de abejas, y una colonia de hormigas, en fin estos comportamientos se dan de manera natural. Estas observaciones han servido de inspiración para estudiar algoritmos que imiten esos comportamientos colaborativos, este se conoce como enjambres inteligentes.

Esta área tiene diversas aplicaciones como en robótica donde estos algoritmos son para investigar el control de brazos robóticos, planeamiento del movimiento del robot, búsqueda colectiva de robots, evasión de obstaculos, swarm robotics, navegación de vehículos no tripulados, y mapeo de regiones.

Entonces el problema que se encuentra la robótica con múltiples robots es definir su coordinación para colaborar en una tarea. Para la coordinación de los robots primero es tratar de indicar la formación que deberán de mantener cuando estén realizando la tarea. Los robots pueden mantener una formación lineal, en filas de 2 etc. La aplicación de tener un sistema de robots (enjambre) en una formación es para facilitar tareas de búsquedas o exploración en espacios

abiertos.

Este trabajo propone crear un enjambre a base de reglas tomadas de enjambres inteligentes, pero utilizando aprendizaje por refuerzo, una sub-área de la inteligencia artificial.

1.1. Antecedentes

Este capítulo proporciona los antecedentes necesarios para entender el tema principal de formaciones y navegación de un conjunto de agentes. Se presentan algunos casos de investigación con ejemplos extraídos de la literatura relacionada al problema. También se explica la manera en la cual fue solucionado cada uno de los resultados al problemas. Primero se explica algoritmos que se diseñaron para recorrer espacios con un agente, luego se revisa la literatura para el problema de formación de robots.

M. Dervisi et. Al [10] realizaron un trabajo donde un robot/agente debe cubrir una superficie. El objetivo del trabajo es que el algoritmo sea ejecutado y pueda concluir de manera satisfactoria la tarea designada, a pesar de los factores externos en el terreno. El algoritmo tiene diferentes aplicaciones como limpiar un área, ayudar en situaciones de emergencia como buscar en una mina colapsada o vigilar una zona. Este trabajo utilizo otros algoritmos para medir su rendimiento:

- Node Counting: Siempre escoge el espacio con el número menor de visitas.
- LRTA*: Mantiene el número de visitas por espacio, esto para actuar como un repelente para los agentes.
- Wagner's: Es un método donde la celda se actualiza si la celda adyacente, donde se moverá el agente, es de menor valor. La celda actual no se actualiza si el valor es mayor a la celda adyacente.
- Thrun's: El algoritmo de Thrun es un método determinístico utilizado para cubrir un área. Este algoritmo es diferente al LRTA*, ya que thrun's garantiza que el valor de la celda se actualiza pero el LRTA* puede sufrir una reducción en su valor si las celdas adyacentes tiene un valor menor a la celda actual.
- Regency: El movimiento del agente es decidido en base al tiempo que fue visitado por última vez.
- Alarm: Este algoritmo ayuda a escoger la siguiente celda adyacente con el mayor número de feromonas.

Como parámetros de experimentación se agrego una variable de saltos. El rendimiento se calculó utilizando las diferentes versiones de algoritmos para cubrir áreas cuando el agente/robot decide saltar y no moverse a la celda adyacente. En los resultados explican que hay un ligero impacto en el rendimiento de los algoritmos que describen como positivo. Los saltos si mejoran el rendimiento del algoritmo pero cuando se encuentra en un terreno más complejo es evidente la caída del rendimiento de los algoritmos.

Otro proyecto desarrollado, trato sobre un vehículo acuatico que recorre la superficie marina [12]. Este trabajo describe un algoritmo lineal para cubrir un terreno desconocido en tres dimensiones bajo el agua. Este algoritmo es para planeamiento de movimientos en un espacio irregular. Este trabajo demuestra que el robot puede recorrer en su totalidad y fotografiar un área sin pasar más de una vez un segmento del área.

El robot esta equipado con sensores para poder recabar la información necesaria para planear la ruta y mantener una distancia del suelo. El robot realiza un movimiento en zigzag usando como guía el campo de visión de la cámara.

El área a recorrer es un rectángulo de dimensiones $lxwxh$ (length x width x height) con el robot al centro. Un problema que resuelve este algoritmo de navegación es detectar islas y otros pequeños espacios. Las áreas más pequeñas son tratadas como sub-problemas, el robot recuerda en donde inician estos espacios, los recorre y al terminar sigue su trayectoria, esto asegura que cada uno sea recorrido una sola vez.

En este siguiente trabajo, realizado por [27], se explica como los robots no tripulados tienen una variedad de aplicaciones en ambientes de incertidumbre y peligro. Estos vehículos pueden ser tele-operados o autónomo. Para describir un vehículo como autónomo (aéreo, acuático o terrestre), este debe incluir lo siguiente:

- Percepción: usar los datos que recolecta del ambiente y de si mismo.
- Inteligencia: Operar sin intervención humana.
- Acción: Viajar del punto A al punto B.

En este artículo utilizan lógica difusa con un vehículo autónomo que para pueda aprender a cuales acciones lo llevan a cumplir su meta. La arquitectura implementada tiene como entrada los datos de los sensores del vehículo aéreo, estos son enviados a un modulo denominado “sensor fusión” el cual le indicara al vehículo de posibles obstáculos. Estos datos son enviados a un controlador de movimientos difuso que también toma como entrada el grado de error de la posición y orientación. Esta arquitectura asegura que el vehículo evada el obstáculo en un

escenario dinámico. Esta arquitectura se probó en diferentes tipos de vehículos autónomos y se demostró ser muy efectiva para realizar su tarea.

En esta sección se hablan de técnicas para la formación de múltiples agentes. En el artículo escrito por [5], utilizan teoría de grafos para el control de movimientos de una formación de robots en una simulación. Los agentes están equipados con sensores para apoyar en la navegación. Se presenta un algoritmo que no necesita a los agentes comunicándose para realizar la tarea en mano. Utiliza un agente como líder para dar la dirección y velocidad de los demás agentes. En los resultados se explica como la formación de diamante y triangular logra converger utilizando el método de control desarrollado. El sistema de control está conformado por diferentes módulos que se encargan de controlar la distancia entre agentes y su líder, que tan rápido se acercan a esos agentes, sistema de corrección de errores. Aun así, se comenta en el artículo que al momento de llegar a su punto de convergencia, tiende a oscilar en el punto dado al sistema laser para el rango de distancia. Una vez que estén juntos, el sistema de robots puede moverse en unisión pero tiene un problema, cuando se acerca a dar una vuelta la agrupación de robots no logra mantenerse.

En otro artículo se prueba la opción de formación utilizando Q-Learning. El trabajo [?] trata de que n-agentes se organicen para llegar a un estado meta que describe la figura a formar vía una ruta óptima. En el método propuesto lo que se intenta realizar es que múltiples agentes y un ambiente denominado “mundo de aprendizaje”, apliquen un método de aprendizaje por refuerzo pero intercambiando información.

En este problema describen un mundo de $N \times N$ con n agentes donde cada agente tiene que llegar a un estado meta (estado G). Cada Agente está habilitado para realizar 5 acciones las cuales son arriba, abajo, derecha, izquierda, y mantener su posición.

El problema es que cada agente logre aprender una póliza que minimice el número de acciones que lo lleva a su meta. Este artículo se basó con la idea de utilizar un método que evalúa la Q no con el número mínimo de acciones sino con un número de acciones que no afecte a los demás agentes, esto es calculado con su varianza y la suma descontada de recompensas.

Una vez evaluado se procede a elegir las mejores Q con la siguiente regla. Si un agente llega a su estado meta con el mínimo número de acciones esto nos indica que tendrá una recompensa más pesada y la varianza es más pequeña. Con esto se realiza una búsqueda de las mejores recompensas y varianzas por episodio en cada mundo de aprendizaje y esas son las que se actualizan para los otros mundos.

Los resultados mostraron que este método es mejor que Q-Learning con esta

configuración de solo sumar el número de pasos a la meta.

En resumen se relataron diferentes ejemplos de investigaciones pasadas sobre temas relacionados al problema que se abarcara en esta tesis. Se explicaron métodos para resolver el problema de recorrer espacios y como se solucionan y algunos de sus problemas. De igual manera se explicaron algoritmos que solucionan la formación de agentes y sus problemas. En la mayoría de los casos se utilizo algún método ya conocido como teoría de grafos o lógica difusa u otros algoritmos. En este caso de la tesis se abordó el problema utilizando aprendizaje maquina y reglas inspiradas con enjambres inteligentes para resolver el problema de formación y posteriormente que recorran un espacio delimitado.

1.2. Organización de la tesis

A continuación se presenta una descripción del contenido de los capítulos 2, 3, 4, 5, y 6. El capítulo 2 contiene el marco teórico donde se explica los conceptos relacionados con el área de investigación. El capítulo 3 trata sobre la metodología donde se proporciona una descripción detallada del desarrollo de las soluciones a los problemas de formación y recorrido en tres dimensiones. El capítulo 4 presenta los resultados experimentales obtenidos con el algoritmo de formación en dos dimensiones, y el recorrido del espacio en dos y tres dimensiones. Finalmente, en el capítulo 5 se presentan algunas conclusiones del trabajo de investigación y recomendaciones para trabajos futuros.

Capítulo 2

Marco teórico

2.1. Aprendizaje Automático

El aprendizaje automático, es una sub área de la Inteligencia Artificial, que se enfoca en desarrollar técnicas de aprendizaje para computadoras. Mucho antes de estas técnicas, los sistemas informáticos estaban diseñados para solucionar problemas predefinidos y por lo tanto eran difíciles de replicar en diferentes escenarios.

Desde hace dos décadas la computación se encuentra en una etapa de lograr en resolver las dificultades del mundo real generando modelos para programas ya que no todo puede ser programado. Así el momento de pasar de programar una infinidad de reglas y ahora enfrentar el reto para lograr que una computadora aprenda de su ambiente, de sus errores y de grandes volúmenes de datos.

Una definición formal de Aprendizaje Automático es “un conjunto de métodos computacionales que utilizan la experiencia para mejorar el rendimiento del mismo o realizar predicciones precisas” [21] . Cuando un sistema utiliza aprendizaje automático el término experiencia se refiere a datos, estos pueden ser generados vía interacciones con el ambiente, y pueden estar etiquetados o no para su análisis

El aprendizaje automático está dividido en tres áreas principales: aprendizaje supervisado, aprendizaje no supervisado, y aprendizaje por refuerzo. En aprendizaje supervisado la tarea es inferir una función partiendo de datos supervisados. Esto quiere decir que los datos utilizados para entrenamiento tendrán una etiqueta. En el aprendizaje no supervisado la entrada es un conjunto de datos no etiquetados, y la tarea consiste en predecir la etiqueta de datos nuevos desconocidos. En aprendizaje por refuerzo la fase de entrenamiento y prueba están mezclados, en este tipo de aprendizaje para obtener información se crea un agente, el cual interactúa con el ambiente y obtiene una recompensa. El objetivo es maximizar su recompensa a través de un período de tiempo en lo que prueba

acciones en el ambiente. Estas acciones pueden ser exploradas o explotadas de acuerdo al conocimiento obtenido [21].

En retrospectiva es evidente que esta área depende mucho de los datos a utilizar. Las técnicas de aprendizaje son métodos que utilizan conceptos de áreas como probabilidad, estadística y optimización. Por lo tanto, en cada escenario es de importancia tener una alta calidad y volumen de datos para el correcto funcionamiento de estas técnicas de predicción o clasificación.

Con el pasara del tiempo muchas areas han encontrado como incluirlo en sus procesos. Aprendizaje automático ayuda a dotar las aplicaciones para ir mejorando en ambientes dinámicos, esto es así ya que utiliza su experiencia para tomar decisiones. Aprendizaje automático tiene diversas aplicaciones en diferentes áreas de la industria.

En el área de la salud se a implementado sistemas de aprendizaje automático para predecir convulsiones y ataques fulminantes [14]. La empresa Healint, localizado en Singapore, creó una aplicación llamada JustShakeIt el cual habilita al usuario mandar alertas de emergencias a sus contactos cercanos con simplemente sacudir el celular. Esta aplicación utiliza un algoritmo de aprendizaje automático para reconocer entre una acción rutinaria y una de emergencia.

Otra aplicación es para la personalización de campañas de marketing [19]. En está área lo que se busca es entender al cliente y ser efectivo al momento de ofrecer servicios de venta de prodcutos en línea, esto ayuda acercarse a cerrar una venta. Este es el concepto general de cómo funciona este tipo de marketing. Hoy en día las empresas tienen mayor conocimiento de los gustos de sus potenciales clientes al analizar sus comportamientos de búsqueda en el internet. Con esta información las empresas pueden saber qué tipos de productos ofertar a traves de los correos electronicos de los clientes o bien si enviar cupones a los domicilios de sus clientes que tienen mayor probabilidad de realizar una comprar con ellos.

De igual manera en el ambito financiero para la detección de Fraude [19].Está área va mejorando día a día en cuanto a detección de fraudes en diferentes áreas. Por ejemplo, PayPal utiliza aprendizaje supervisado para luchar contra el lavado de dinero, ya que cuentan con las herramientas para detectar una transacción fraudulenta y un comprador o vendedor legitimo.

2.2. Aprendizaje por Refuerzo

El aprendizaje por refuerzo es aprender qué hacer (cómo relacionar situaciones con acciones) para maximizar (o minimizar) una señal numérica de recompensa [26]. Este tipo de aprendizaje no está caracterizado por el algoritmo a utilizar, pero se define por el tipo de problema donde se aplica el aprendizaje. En cualquier

caso se define un ente de aprendizaje, el cual posee características muy simples, éste se conoce como un agente.

En un escenario de aprendizaje por refuerzo el agente no recibe un conjunto de datos etiquetados, el proceso de recabado de datos es que el agente realizará acciones para interactuar con el ambiente y, como respuesta, recibir una señal numérica (recompensa) y estado actual. El objetivo del agente es maximizar el rendimiento esperado, el cual se define como la suma esperada de las futuras recompensas recibidas del ambiente. Esto se hace mediante el aprendizaje de una política que relaciona estados con acciones.

El aprendizaje por refuerzo se encuentra entre el aprendizaje supervisado y el aprendizaje no supervisado. Una de las mayores diferencias entre éstos es que el agente (aprendiz por refuerzo) debe explorar explícitamente su ambiente [16]. Los problemas de aprendizaje por refuerzo son comúnmente formulados como procesos de decisión Markov (MDP, por su siglas en inglés).

2.2.1. Procesos de Decisión Markov (MDP)

Un MDP es un proceso Markoviano, el cual formalmente es representado como una cuádrupla $\langle S, A, P, \gamma, R \rangle$ [8], donde :

- S es el conjunto de estados
- A el conjunto de acciones
- $P(s_{t+1}|s_t, a_t)$ es una función de transición que relaciona las transiciones entre estados $s_t, s_{t+1} \in S$ dada una acción $a \in A$
- R es la función de recompensa que asigna los pares estado-acción con números reales $r' = r(s, a)$.

El proceso se describe como Markov por que la función de transición y de recompensa depende del estado actual s y no de la historia de estados y acciones tomadas. Es decir que un agente se encuentra en un estado $s_t \in S$ selecciona una acción a_t del conjunto A . Esto lleva al agente a recibir una recompensa de valor esperado $R(s_t, a_t)$, esto da como resultado un cambio de estado de acuerdo a la función de transición $P(s_{t+1}|s_t, a_t)$. Una vez en el siguiente estado s_{t+1} el agente escoge una nueva acción a_{t+1} , por lo tanto la recompensa $R(s_{t+1}, a_{t+1})$ y así al siguiente estado s_{t+2} .

Por lo tanto el proceso de transición se representa de la siguiente manera

$$S_0 \xrightarrow{a_0} S_1 \xrightarrow{a_1} S_2 \xrightarrow{a_3} S_3.$$

Ahora las recompensas se van acumulando al cambiar de estados s_t y obtiene las recompensas:

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots$$

El objetivo del MDP es maximizar la suma de las recompensas esperadas por el agente. Para esto se define una política $\pi : S \rightarrow A$ para mapear los estados a acciones. A su vez se define una función de valor V para una política π como la suma esperada con descuento que son obtenidas cuando se ejecuta un acción dada por π . Esto se ve en la siguiente formula

$$V^\pi = E[R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots | s_0 = s, \pi] \quad (2.1)$$

Entonces V^π es la suma de recompensas con descuento que recibira el agente si comenzara en el estado s y las acciones dadas por la política π . Esta es la ecuación de Bellman.

$$V^\pi = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V^\pi(s') \quad (2.2)$$

Para obtener la función de valor maxima e define como:

$$V^* = \max_{\pi} V^\pi(s) \quad (2.3)$$

La ecuación óptima de Bellman se define como:

$$V^\pi = \max_{a \in A} \left[R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V^\pi(s') \right] \quad (2.4)$$

2.2.2. Elementos del aprendizaje por refuerzo

En este tipo de problemas, es posible identificar cuatro componentes esenciales para el sistema de aprendizaje. Estos son: la política, la función de recompensa, la función de evaluación, y el modelo del ambiente [26]. En la siguiente figura ?? se ve un diagrama de las partes de un sistema de aprendizaje por refuerzo.

- **Agente:** Es un ente que debe aprender un forma de operar para cumplir su objetivo. Este objetivo es maximizar el total de recompensa obtenida dentro del ambiente.
- **Ambiente:** Un sistema de aprendizaje por refuerzo es una representación de estado-acción por interacciones de prueba y error en el medio. El agente con el pasar del tiempo debe poder observar lo que esta en su estado actual. Esto es así ya que el sistema de aprendizaje le dara la información requerida para realizar una acción.

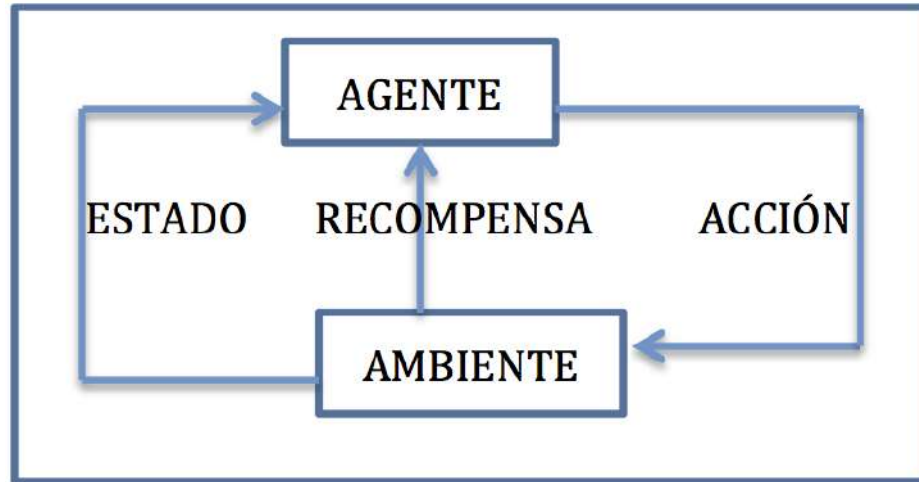


Figura 2.1: Diagrama de aprendizaje por refuerzo

- **Política:** De acuerdo a la literatura [26], este define el comportamiento del agente en un tiempo. Lo describen como un mapeo de los estados del agente y las acciones a elegir en los mismos. Esto en el área de psicología se conoce como reglas de estímulo-respuesta o asociaciones. La política π es importante para definir el comportamiento del agente.
- **Función de recompensa:** El problema de aprendizaje por refuerzo, al mapear cada estado-acción del agente a valor numérico (recompensa R) este nos dice que tan bueno es el estado siguiente. El agente tiene como objetivo maximizar este valor numérico a largo plazo, de tal manera la función dicta que acción estado son bueno y cuales son malos. Esta función es inalterable durante la ejecución del algoritmo pero ayuda a la alteración de la política, por ejemplo, cuando el agente en un estado elige un acción con muy baja recompensa, la política debe cambiarse para elegir una acción diferente para ese estado.
- **Función de evaluación:** Como se describe con la función de recompensa, este indica lo que es bueno en lo inmediato, la función de evaluación nos dira que es bueno a largo plazo. La función de evaluación dice el valor total que puede recibir un agente en un estado inicial hasta su estado final.

2.2.3. Metas y recompensas

El agente recibe una señal numérica, $r_t \in R$, en cada paso que haga. El agente debe maximizar la recompensa acumulada que recibe. Un ejemplo en la literatura, es si un agente se mueve recibe -1, y 100 si llega a la meta. Esto hara que el robot trate de llegar a la recompensa de 100 lo más pronto posible. El proposito de la

recompensa es que el agente tenga el conocimiento que debe hacer y no como debe hacerlo. El retorno se define como la secuencia de recompensas. Como se explicó el agente recibe una recompensa r_{t+1} dado un estado $s_t \in S$ y una acción $a_t \in A(s_t)$ y se mueve a un nuevo estado s_{t+1} .

Las recompensas después de un tiempo t se denotan como $r_{t+1}, r_{t+2}, r_{t+3}, r_{t+4} + \dots + r_n$ lo que se busca es maximiar el retorno (R_t), el cual se da con la siguiente formulación:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + r_{t+4} + \dots + r_T \quad (2.5)$$

Donde T es el tiempo final, donde si es finito este se denomina tareas episódicas, si es infinito se conoce como tareas continuas. Por lo tanto la forma anterior no indica un límite. para eso se da la siguiente formulación:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (2.6)$$

El parámetro *gamma* se conoce como la tasa de descuento y representa el valor presente de la recompensa futura y tiene valores $0 \leq \gamma \leq 1$. Si $\gamma = 0$ el agente solo aprende de la recompensa inmediata (acción $t+1$). Cuando $\gamma > 0$, la recompensa fue recibida un tiempo en el futuro k , se pondera por γ^{k-1} , esto significa que el agente recibe una fracción de lo hubiera recibido. Por lo tanto mientras que γ tienda a 1, las recompensas futuras tienen más efecto en el agente [26].

2.2.4. Estrategias de exploración-explotación

En un problema de aprendizaje por refuerzo, por lo general el agente tiene cero conocimiento de la tarea aprender, no conoce las funciones de transición y recompensa. El agente tiene un desconocimiento total de lo que se requiere realizar. Por lo tanto el agente no sabe la secuencia de acciones que lo haran obtener el mayor retorno posible, entonces lo primero que hara el agente es explorar.

En aprendizaje por refuerzo el agente se le permite realizar una exploración de los estados y acciones que puede realizar para que pueda descubrir cuales le dan buenas recompensas a corto y largo plazo. Una vez que el agente a realizado una exploración total, puede explotar su conocimiento.

El dilemma de la exploración-explotación es encontrar un balance donde el explorar mucho no degrade el sistema de aprendizaje, o bien una explotación temprana de resultados no óptimos. Una estrategia muy conocida es ϵ -greedy, el cual define una probabilidad ϵ de ejecuta acciones de forma avariciosa, y una probabilidad de $1 - \epsilon$ ejecutando acciones de forma aleatoria [?].

2.2.5. Q-Learning

Uno de los métodos más comunes de aprendizaje por refuerzo es Q-Learning. Para cada par (s, a) el algoritmo mantiene un registro del promedio actual de recompensas r , el cual recibe al dejar el estado s con una acción a , adicionalmente con su recompensa esperada.

Este algoritmo con suficientes iteraciones puede lograr converger a un Valor-Q óptimo. Este se llama un algoritmo “off-policy”, la póliza de entrenada no es la que se esta ejecutando, el cual da entender que con sólo ir moviendo el agente con acciones aleatorias logra aprender la póliza.

Como se mencionó, el algoritmo nos garantiza una convergencia si el agente logra visitar cada estado y cada transición las veces necesarias. La gran desventaja que es inherente en este algoritmo es que el tiempo de entrenamiento puede ser tardado.

Para lograr reducir los tiempos de entrenamiento se opta por implementar una estrategia conocida como ϵ -greedy. E-greedy es una estrategia para escoger una acción de manera aleatoria o codiciosa (escogiendo el mejor Valor-Q para la acción del estado) con una probabilidad de $1-\epsilon$. Implementando esto, podemos garantizar que más tiempo explore las secciones interesantes del ambiente así mejorando el Valor-Q pero seguirá visitando áreas no conocidas. En un proceso de aprendizaje es normal que la ϵ inicie con un valor de 1 y gradualmente vaya disminuyendo hacia 0.05.

La forma más simple de Q-Learning se define en la ecuación 2.7.

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \min_a Q(s', a') - Q(s, a)] \quad (2.7)$$

Q-Learning estima una función de acción-valor $Q^\pi(s, a)$, con $s \in S$ y $a \in A$ para una política π que asigna los pares estado-acción a valores escalares que representan el rendimiento esperado de tomar una acción a en el estado s y partir de allí actúa de acuerdo a una política π . En este caso, la función de acción-valor aprendida, $Q(s, a)$, aproxima directamente $Q^*(s, a)$, la función acción-valor óptima, independientemente de la política que se esté siguiendo. Lo anterior se representa en la figura ??.

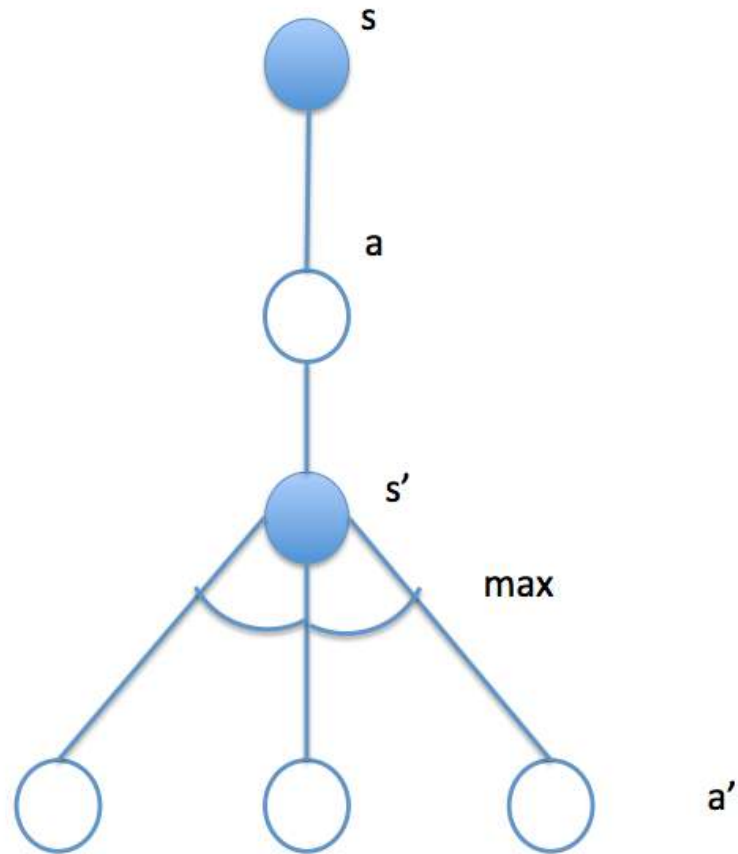


Figura 2.2: Diagrama de apoyo Q-Learning

Q-learning es un método muy utilizado para la resolución de problemas de aprendizaje por refuerzo, esto dado a su simplicidad de implementación. A continuación, se presenta el algoritmo de Q-Learning:

Algorithm 1 Algoritmo de Aprendizaje Q-Learning

```

1: function Q-LEARNING( $\pi$ )
2:    $Q \leftarrow Q_0$  inicialización, ej  $Q_0 = 0$ 
3:   for  $t \leftarrow 0$  to  $T$  do
4:      $s \leftarrow \text{elegirEstado}()$ 
5:     for cada paso de episodio do
6:        $a \leftarrow \text{SelectAccion}(\pi, s)$  póliza  $\pi$  derivada de Q, ej,  $\epsilon$ -greedy
7:        $r' \leftarrow \text{Recompensa}(s, a)$ 
8:        $s' \leftarrow \text{NextState}(s, a)$ 
9:        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \min_{a'} Q(s', a') - Q(s, a)]$ 
10:       $s \leftarrow s'$ 
11:  return Q

```

En cada episodio una acción es seleccionada en el estado actual s utilizando la póliza π derivada de Q . La elección de esta póliza es arbitraria tanto así que garantiza que el agente visite cada par (s, a) infinitas veces.

Aprendizaje por refuerzo se ha aplicado con éxito en diversas áreas de la industria, a continuación se describen algunas aplicaciones de algoritmos de esta área.

En Tokio, la fabrica Fanuc creo un robot utilizando aprendizaje por refuerzo para aprender tareas. Al robot se le asigna una tarea por ejemplo de tomar piezas de una caja y colocar esas piezas en otra caja, esto lo estará realizando toda la noche hasta que lo aprenda [18].

El robot de la empresa Fanuc utiliza una técnica como deep reinforcement learning para el entrenamiento. El robot intenta levantar un objeto mientras captura video del proceso. Esto es para que en el momento que falle o se exitoso pueda diferenciar utilizando la posición del objeto en ese momento[18].

Así como es aplicado en robótica, se pueden encontrar problemas donde se necesita un agente de aprendizaje en un ambiente cambiante. Asignar precios de manera dinámica depende mucho sobre la demanda y abasto para maximizar la ganancia de los productos. Técnicas como Q-learning puede utilizarse para proveer una solución a este problema. Aprendizaje por refuerzo ayuda en optimizar precios durante su interacción con el cliente.

2.3. Enjambres Inteligentes

En la actualidad, la naturaleza se ha vuelto un medio de inspiración para el área de computación. Un ejemplo de esto es el comportamiento colectivo de los animales o insectos [17]. Considere una colonia de hormigas, un enjambre de abejas, una colonia de bacterias, y una bandada de estorninos, en estos casos todas estas agrupaciones funcionan con reglas muy sencillas para mantenerse organizados y cumplir un objetivo. Cada elemento o agente que forma parte de la agrupación, por si solo no es capaz de realizar el objetivo. Entonces, al estar en una agrupación, es más fácil la adaptación al ambiente para los agentes, ya que cada agente recaba información que ayuda en el desempeño del enjambre.

Enjambres inteligentes es un comportamiento colectivo, descentralizado de sistemas naturales o artificiales. Este concepto fue introducido por Gerado Beni y Xing Wang en 1989 [17]. El termino enjambres es utilizado para hacer referencia a los diversos tipos de colecciones de agentes.

Estos algoritmos son rápidas y robustas soluciones para resolver problemas complejos. Enjambres Inteligentes es una nueva rama de la inteligencia artificial que es utilizado para comportamientos colectivos que ocurren en la naturaleza.

Estos agentes simples tiene capacidades limitadas, por lo tanto al apoyarse de otros agentes de su mismo tipo se puede lograr el objetivo. Estos agentes siguen reglas sencillas. La interacción entre los agentes puede ser de manera directa o indirecta. Interacción directa puede ser a través de sonidos, y una interacción indirecta es que el agente realice un cambio en el ambiente para que los otros agentes respondan a ese cambio.

Enjambres inteligentes funcionan con dos principios básicos: auto-organización y stigmergia.

Un sistema de auto-organización está caracterizado por tres parámetros:

- Estructura - el sistema es homogéneo
- Estabilidad - el sistema puede coexistir en más de un estado
- Cambios de estado - el sistema puede cambiar de manera inoportuna si es necesario

Stigmergia es el principio donde un agente utiliza su ambiente para comunicarse para influir en las acciones de sus demás agentes. Es una colaboración por medio del ambiente. Estas son sus características:

- Funciona como un agente indirecto para en el ambiente
- Lo que esta en el ambiente funciona como memoria externa para el agente
- El objetivo puede ser terminado por cualquier agente

Para que un enjambre pueda ser declarado inteligente, este tiene que cumplir con 4 principios como fue enunciado por Mishra et al. [?] y Keerthi et al. [17]:

- Principio de proximidad: los agentes que componen el enjambre deben de ser capaces de realizar computos en base a su interacción en el ambiente.
- Principio de calidad: deben de poder responder a los factores que influyen en su ambiente. Estos factores son comida y su seguridad.
- Principio de respuesta diversa: esto se refiere a que el enjambre sabe dividir sus recursos entre todos los agentes para protegerse de momentos adversos
- Principio de adaptación: el enjambre debe de ser capaz de sobrevivir momentos de peligro en su ambiente, de tal manera debe de cambiar su forma de funcionamiento al instante tomando en cuenta el desgaste de energia.

2.3.1. Optimización de enjambre de partículas (PSO):

La optimización de enjambre de partículas es un método presentado por el Dr. Kennedy y Berhart en 1995 [11]. Este método en particular fue inspirado por el comportamiento colectivo de las aves y los peces. Normalmente estos animales están viajando en busca de comida. Mientras el enjambre viaja habra un individuo que detecte la comida y guiara a los demas para acercarse al objetivo.

Por ejemplo las aves se comunican a través de sonidos para dirigirse y tienen una formación en V. La formación en V de las aves tiene dos teorías. La primera teoría es que al momento de volar, el aire que se va desplazando de las alas se va hacia atrás y esto le da una ayuda a la ave de atrás para realizar un menor esfuerzo. Una vez que se tenga el movimiento el pájaro que está guiando va rotando su posición con los demás, esto ayuda a maximizar el esfuerzo de cada pájaro, para que así puedan volar por un período más largo sin tomar un descanso. La segunda teoría es que su formación se debe a que quieren poder ver todos los elementos de la bandada.

En contraste al algoritmo anterior que utiliza un método indirecto, estigmergia, para poder llegar a la solución, este va manteniendo un estado global para todo el enjambre, comunicación directa. El estado global es el valor más cercano al objetivo, que sera el pajaro en la punta de la formación.

Un modelo sencillo de enjambre de partículas se describe con estas tres reglas:

- Los vecinos se mueven en la misma dirección.
- Se mantienen cerca de sus vecinos.
- Evitan colisiones.

Cada partícula mantiene sus coordenadas del espacio de búsqueda, y lo relacionan a su mejor vuelo ($pbest$), su experiencia de vuelo de la partícula. Otro valor que las partículas toman en cuenta es el mejor valor de sus vecinos ($lbest$). Así como cada partícula va checando sus vecinos hay un valor global de referencia para todas las partículas. Este valor global se conoce como el $gbest$, que es el valor más cercano a la solución [17].

Cada partícula consiste de 3 partes:

- Los datos que representan una posible solución. Esto ya puede ser sus coordenadas. Cada partícula actualizara sus coordenadas a la partícula que esté más cercana a la solución.
- La velocidad de la partícula. Este indica cuanto debe de cambiar para poder alcanzar al $gbest$ del enjambre.

- Cada partícula tiene su atributo *pbest*. Este indica que tan cerca ha llegado la partícula al *gbest*.

Este algoritmo a tenido éxito en diferentes áreas, ya que ha sido demostrado que obtiene mejores resultados, en menos tiempo en comparación a otros métodos. El algoritmo original de partícula de enjambres es el siguiente [11].

1. Inicializar las partículas
2. Para cada partícula evaluar el objetivo
3. En cada iteración la partícula evalúa su mejor valor personal (*pbest*)
4. Comparar el valor de la partícula con el mejor valor del enjambre (*gbest*)
5. Cambiar la velocidad y posición en cada partícula
6. Iterar a partir del paso 2, hasta que se cumpla el objetivo

Esté algoritmo tiene una variedad de aplicaciones, así de forma general o hacer alguna modificación al algoritmo para realizar una labor en específico. En está sección se describirán de manera breve algunas áreas de aplicación [23]:

- Robótica: En esta área se incluyen los siguientes temas más investigados como control de brazos roboticos, planeamiento del movimiento del robot, búsqueda colectiva de robots, evasión de obstaculos, swarm robotics, navegación de vehiculos no tripulados, y mapeo de regiones.
- Redes de sensores inalámbricas: esté algoritmo se aplica para la planeación de las diversas topologías que pueden eficientar la red, y distribución de los sensores.
- Sistemas de predicciones: para predecir la calidad del agua, predicciones meteorológicas, predecir el flujo del tráfico en zonas urbanas.

Es notable que hay más áreas donde se ha aplicado este algoritmo, como problemas de optimización, optimización multi-objetivo, reconocimiento de patrones, problemas de *job scheduling*, planeamiento de rutas en tiempo real, segmentación de imágenes por nombrar algunos.

Capítulo 3

Metodología

En el presente capítulo se describe el procedimiento que se empleará para llevar a cabo el desarrollo de los algoritmos descritos en el capítulo anterior. El problema a resolver está dividido de la siguiente manera:

- (1) Organización de agentes en un espacio discreto
- (2) Algoritmo para cubrir una superficie, utilizando un ente compuesto de agentes
- (3) Crear un ambiente visual para ver el funcionamiento de los agentes virtuales.

3.1. Problema de Formación

En primera instancia se requiere la inicialización de posiciones aleatorias para cada agente local AL en el mundo de cuadrícula, ver 3.1. Los agentes se mueven de sus posiciones iniciales, y su formación se cumple cuando cada agente llega a la posición meta. El número de posiciones meta es igual al número de agentes presente en el espacio discreto, y cada agente llega a una posición distinta a los demás. Este problema se enfoca en que cada agente llegue a su posición meta en el mínimo número de pasos.



Figura 3.1: Diagrama del mundo discreto, donde G es la posición meta y cada agente tiene una posición inicial.

En este problema el mundo se describe como $N \times M$. Donde I , es el número de agentes en el mundo. Cada agente $AL_i (i = 1, 2, 3, \dots, I)$ se desplaza de su posición inicial P_i para llegar a una de las G metas predefinidas. El vector de meta esta representado de la siguiente manera $G = \{G_1, G_2, \dots, G_I\}$. Para este problema el agente tiene 5 grados de libertad, es decir un conjunto de acciones A , mover hacia el frente, atrás, derecha, izquierda, o mantener su posición actual, ver Figura 3.2. Un criterio incluido es que cada agente no puede ocupar el espacio donde se encuentra otro agente.

Este problema se describe como un proceso de aprendizaje por refuerzo. Cada agente tiene la habilidad de percibir su estado actual que esta representado por un índice lineal i que luego se convierte a una coordenada (x_i, y_i) , esto es verdad para todos los agentes. Un estado S para el problema esta representado por la concatenación de las posiciones de cada agente como se describe en la ecuación 3.1. En este caso existe $I!$ de estados.

$$S = AL_i + AL_{i+1} + \dots + AL_n \quad (3.1)$$

Este modelo de estado se adopto para poder identificar un estado de manera unica. Por lo tanto un estado S tiene una cantidad de $a_i! + a_{i+1}! + a_{i+2}! + \dots + a_{i+A}!$ acciones que evaluar para el conjunto de agentes.

Como se menciono al principio, cada agente se puede mover de la siguiente manera:

Movimiento a la izquierda (x_i, y_{i+1})

Movimiento a la derecha (x_i, y_{i-1})

Movimiento hacia atras (x_{i-1}, y_i)

Movimiento hacia adelante (x_{i+1}, y_i)

Movimiento es mantenerse (x_i, y_i) .

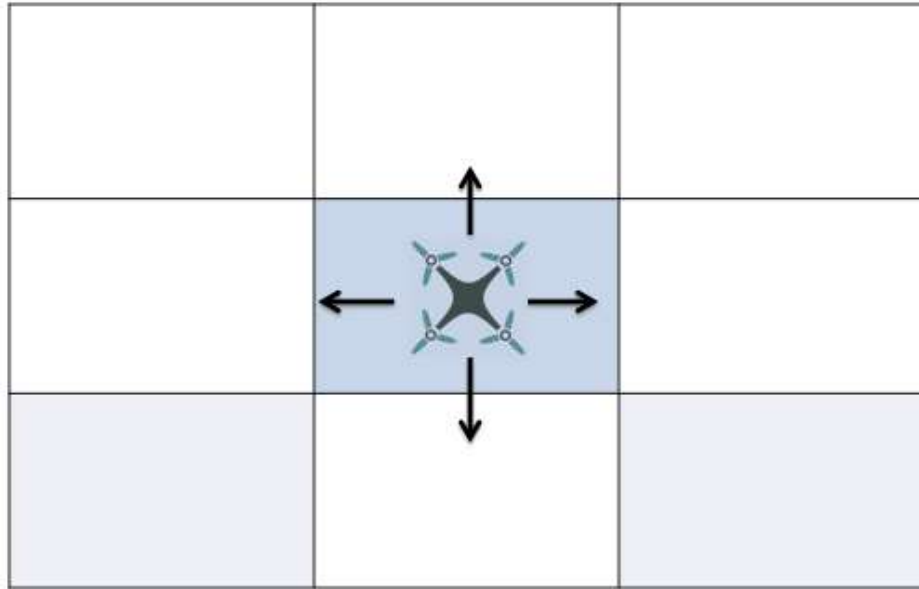


Figura 3.2: Diagrama del mundo discreto, donde G es la posición meta y cada agente tiene una posición inicial.

Cada agente debe llegar a una posición meta distinta a los demás, el conjunto de metas no son entregadas como conocimiento a priori, este tiene que descubrir mediante exploración como llegar a ello a través de su aprendizaje. Con la configuración anterior, el problema de aprendizaje por refuerzo es que cada agente aprenda los movimientos correctos para llegar a una meta G . Para obtener la póliza π optima, al momento de que cada agente llega a una coordenada meta se le otorga una recompensa r de 0, de lo contrario se le entrega una recompensa de -1 cuando logra moverse, si un espacio esta ocupado la recompensa es de -75 .

Los pasos a seguir para la formación de agentes

Paso 1. Se inicializan agentes en posiciones aleatorias en el mundo $N \times M$

Paso 2. El conjunto de agentes puede decidir explorar o seleccionar sus mejores acciones usando epsilon-greedy, ϵ .

Paso 3. Crear el estado s , el cual se mapea a un vector de acciones a tomadas por cada agente.

Paso 4. Cada agente interactúa con el mundo para actualizar los valores de $Q(s, a)$, ver ecuación 2.7. La recompensa para actualizar Q es la sumatoria de las recompensas de cada agente al ejecutar una acción a , ver ecuación 3.2.

Paso 5. Los agentes se mueven hasta que encuentren una posición meta G .

Paso 6. Regresar al paso 1, repetir E episodios. Cuando todos llegan a una posición meta termina el episodio

$$R = \sum_{i=0}^I r_i \quad (3.2)$$

Este algoritmo ayuda para crear diferentes configuraciones con los agentes, en espacios de $N \times M$ con una cantidad de I agentes. Cada formación es guardada en un archivo de texto para luego visualizar en un entorno virtual. Esto es así porque también hay que considerar que se vuelve más complejo el problema, entonces dividir por entrenamientos cada formación se optó como una estrategia al abarcar el problema.

3.2. Cubrir área discreta

Para resolver el problema, se diseñó un algoritmo siguiendo las características de aprendizaje por refuerzo. A continuación se explica el diseño de los estados, la función de recompensa y transición.

Como el problema de formación, aquí se adaptó el conjunto de agentes como uno solo. Este nuevo agente, conformado por agentes locales, tiene los mismos movimientos que estos, ver figura 3.3, el global tiene 5 grados de libertad, es decir un conjunto de acciones AG , mover al frente, atrás, derecha, izquierda, o mantener su posición actual.

Un criterio incluido es que cada agente no puede ocupar el espacio donde se encuentra otro agente. Es decir que una vez se tenga la formación deseada, se procede a mover el conjunto en la dirección que indica el conocimiento del mundo global. El mundo global está compuesto por sub grids del tamaño del grid utilizado para entrenamiento de la formación.

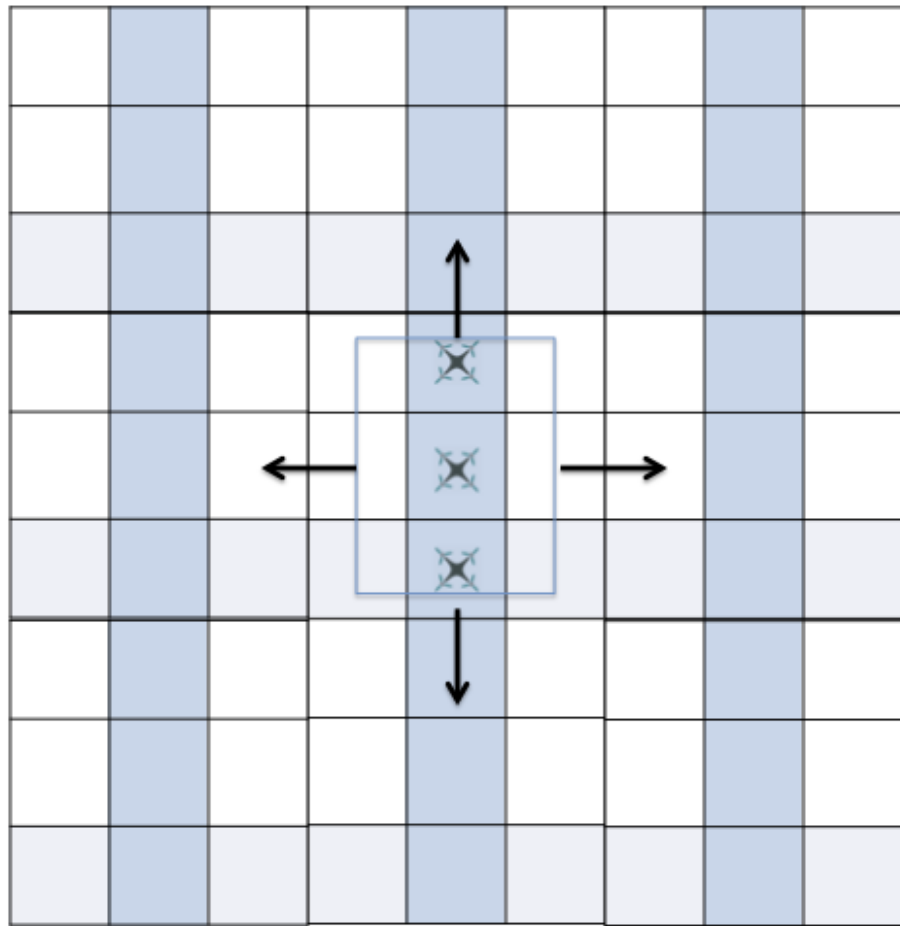


Figura 3.3: Diagrama del mundo discreto, donde G es la posición meta y cada agente tiene una posición inicial.

Para obtener las acciones necesarias que nos ayudaran a recorrer el mundo global se necesita definir los estados del mundo. En este caso un estado es representado por la posición del agente global AG y la concatenación de si un espacio en la cuadrícula fue visitado o no visitado, ver ecuación 3.3. En la ecuación estamos representando un estado para un mundo de 3×3 . De esta manera el estado es único y mapeado a un vector de acciones.

$$S = AG_i + 100000000 \quad (3.3)$$

Los pasos a seguir para que el agente global explore el mundo

Paso 1. Se inicializa el agente en una posición fija en el mundo global $N \times M$

Paso 2. El agente puede decidir explorar o seleccionar sus mejores acciones usando epsilon-greedy, ϵ .

Paso 3. Crear el estado s , el cual se mapea a un vector de acciones a tomadas por el agente.

Paso 4. Cada agente interactúa con el mundo para actualizar los valores de $Q(s, a)$, ver ecuación 2.7. La recompensa para actualizar Q es la sumatoria de recompensa cuando ejecuta una acción y los estados que faltan por visitar, ver ecuación 3.4.

Paso 5. Regresar al paso 1, repetir E episodios. Cuando el agente cubra todo el mundo el episodio termina.

$$R = r_{ag_i} + \text{espacios_por_visitar} \quad (3.4)$$

Este algoritmo recorre todo el mundo de $N \times M$ con un ente compuesto de agentes. Cada formación es guardada en un archivo de texto para luego visualizar en un entorno virtual. Esto es así por que con las formaciones ya entrenadas podemos mover el ente y reorganizar cada agente al momento de cambiar de dirección en el mundo, ver Figura 3.4.

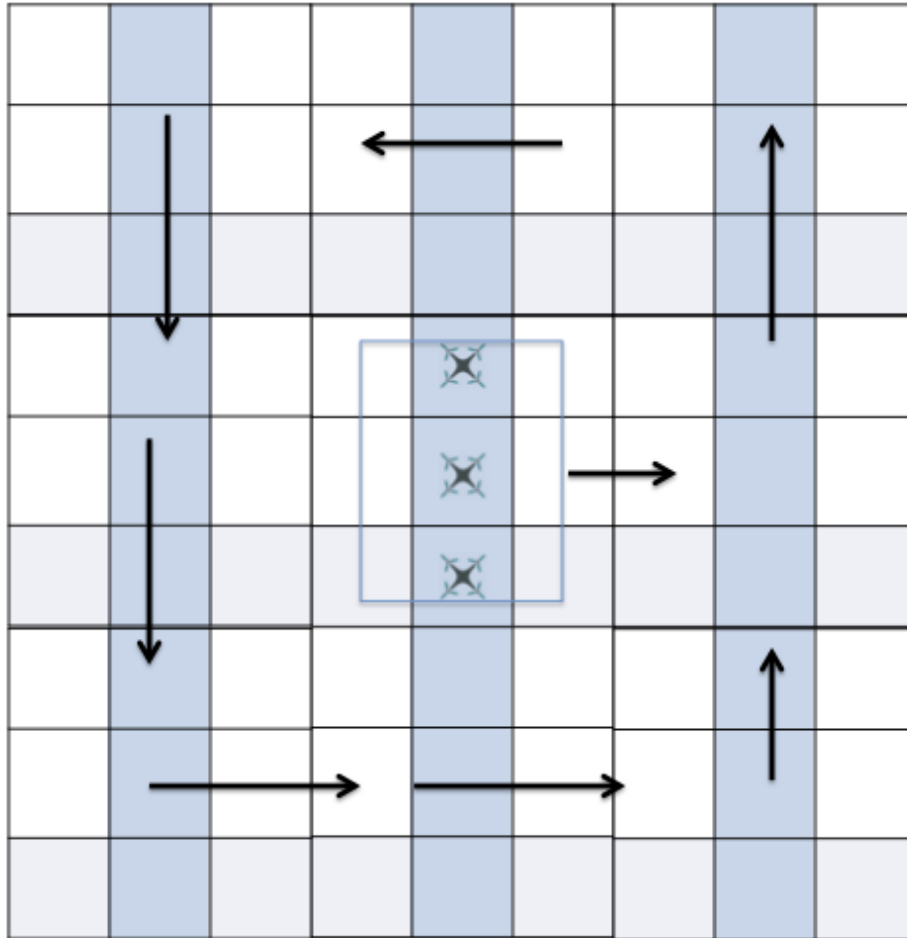


Figura 3.4: Diagrama del mundo discreto, donde el agente global se mueve en las cuadrículas, del mundo global (3×3), y cada cuadrícula local es 3×3 .

3.3. Agentes en mundo de 3-Dimensiones

Extendiendo el problema de 2 a 3 dimensiones se a modelado de la siguiente manera:

El mundo se compone de ejes (x, y, z) , donde $x = n, y = m, z = 3$. Esto nos indica que se cuentan con tres niveles de altura para el desplazamiento del agente, ver figura 3.5 .

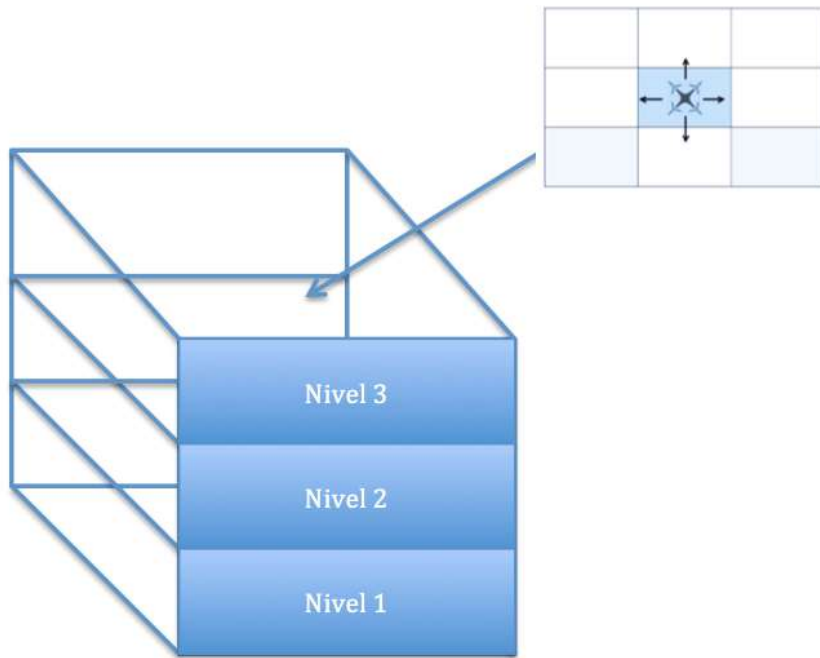


Figura 3.5: Representación de un espacio en mundo tri-dimensional. Esta compuesto por un grid de $n \times m$ en los tres niveles y solo representa una celda del mundo.

Como en el problema anterior el agente tiene que llegar a su posición meta G . Para este problema el agente tiene 2 movimientos además de los ya explicados, sus acciones A , mover hacia el frente, atrás, derecha, izquierda, arriba, abajo o mantener su posición actual, ver Figura 3.6. Igual que los problemas anteriores, el criterio que cada agente no puede ocupar el espacio donde se encuentra otro agente se mantiene.

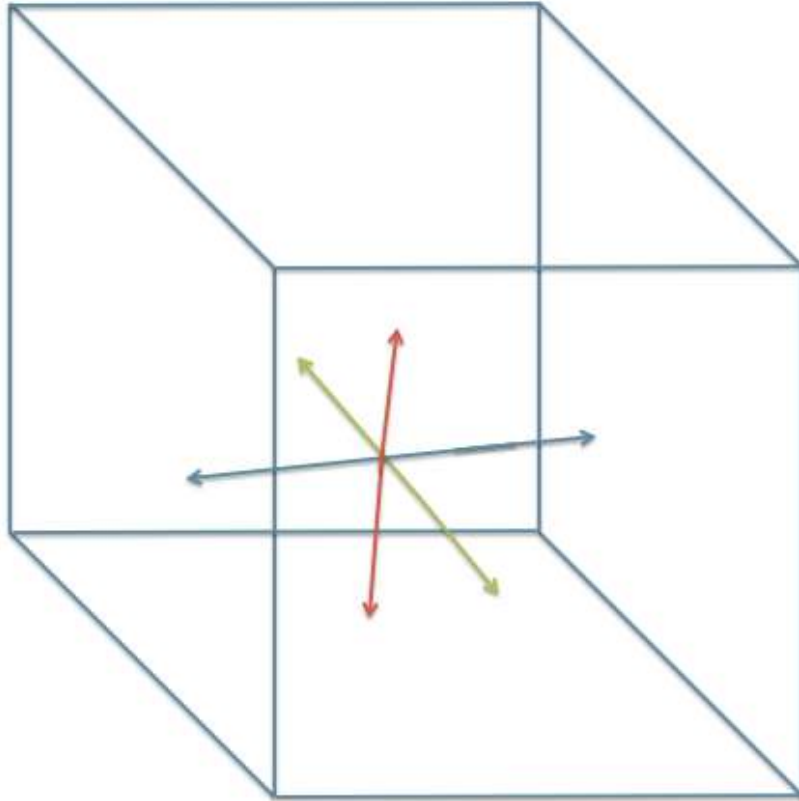


Figura 3.6: Representación de los movimientos de un multi-agente en un espacio del mundo en tres dimensiones.

Para que el algoritmo logre aprender a navegar este espacio y encuentre sus posiciones meta, se tiene que describir el estado y las recompensas correspondientes.

En este caso un estado es representado por la posición del agente global AG y la concatenación de si un espacio en la cuadrícula fue visitado o no visitado a de mas del nivel donde se encuentra, ver ecuación 3.5. En la ecuación estamos representando un estado para un mundo de 3×3 . De esta manera el estado es único y mapeado a un vector de acciones.

$$S = AG_i + 100000000 + z \quad (3.5)$$

En cuanto a la recompensa para este mundo, se penaliza -1 cuando se mueve a un espacio libre (r_{ag_i}), se penaliza -50 cuando esta en la parte inferior del mundo o superior del mundo ($nivel$), se penaliza 0 cuando se mantiene en el segundo nivel del mundo y llega su posición meta.

$$R = r_{ag_i} + nivel \quad (3.6)$$

3.4. Simulaciones

Para poder demostrar que los algoritmos funcionan de manera satisfactoria, además de realizar experimentos numéricos, se desarrollaron dos ambientes para realizar pruebas visuales de dos dimensiones y tres dimensiones. Ambos programas leen un archivo de entrada donde se indica el número de agentes a utilizar, las dimensiones del ambiente y las acciones a realizar. Para el desarrollo de un ambiente en dos dimensiones se utilizó la librería *pygame*, una librería para el lenguaje python. La librería es en su mayoría utilizada para crear animaciones y juegos en dos dimensiones. Para propósitos de esta tesis se implementó para crear mundos de $n \times n$. En la siguiente Figura 3.7 se tiene un mundo de cuadrícula.

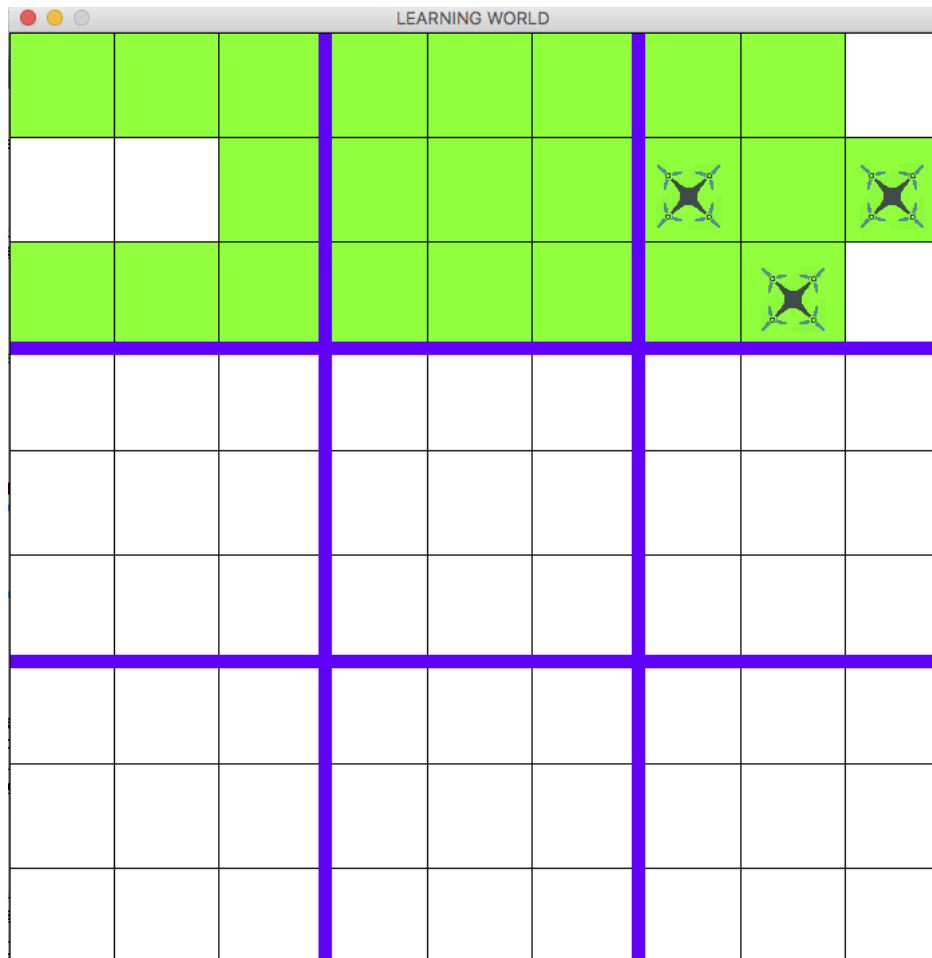


Figura 3.7: Figura de un mundo de cuadrícula de 3×3

Para las pruebas de tres dimensiones se desarrollo un ambiente con la libreria de javascript *three.js*. Three.js es usado para crear y dibujar animaciones en tres dimensiones en los navegadores web, utiliza tecnología WebGL. En este caso se implemento con dimensiones de $n \times n \times n$. En la siguiente Figura 3.8 se tiene un mundo de 3 dimensiones.

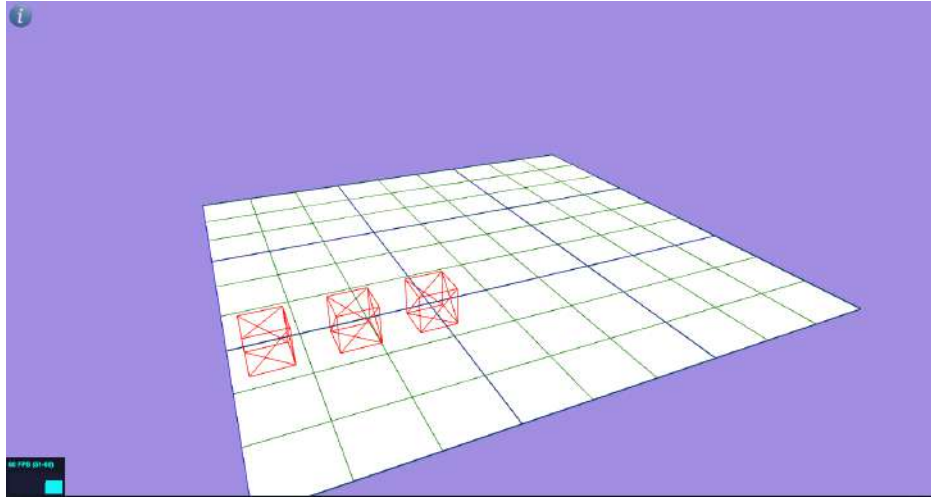


Figura 3.8: Figura de un mundo de cuadrícula en 3 dimensiones con 3 representaciones de agentes

3.4.1. Funcionamiento general de las simulaciones

Básicamente, el funcionamiento de ambas simulaciones hace uso de las soluciones para su funcionamiento. De acuerdo a la formación se genera un conocimiento de acciones. Por ejemplo para la formación de tres agentes, ver figura 4.5 en el capítulo 4, se generan cuatro formaciones validas de acuerdo a las acciones que puede elegir el agente, estas acciones son derecha, izquierda, adelante, regresar. Entonces el multi-agente con estas acciones saber si es necesario utilizar el conocimiento para lograr el objetivo.

Los pasos principales del algoritmo para la simulación se ilustran en la figura 3.9 y se describen a continuación

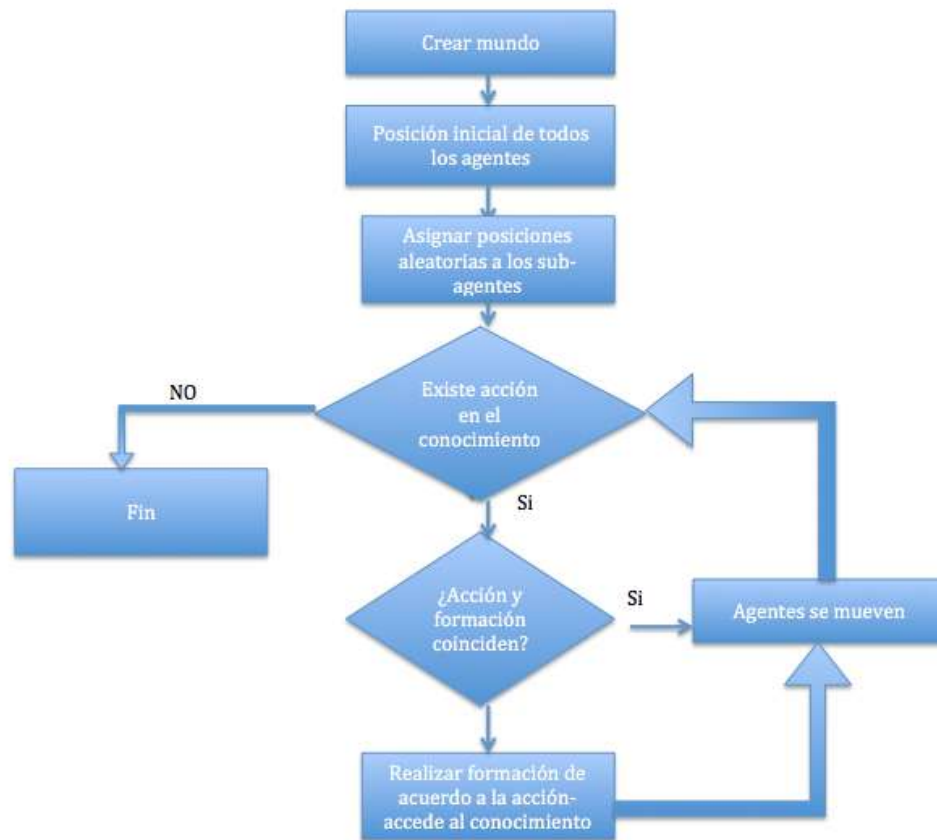


Figura 3.9: Diagrama de flujo de simulación.

1. Se crea un mundo de $NxNxN$ o de NxN dependiendo la simulación.
2. Se elige la posición inicial del agente.
3. Se eligen posiciones aleatorias para los sub agentes
4. Elige una acción. Si el agente no esta en la formación de acuerdo a la acción elegida procede a formarse y luego se mueve. De lo contrario si ya esta formado y coincide con la acción procede solo a moverse.
5. 6El agente repite el paso 4 hasta no tener más acciones y cubra todo el espacio.

3.5. Resumen

En este capítulo, se explicaron los aspectos básicos de los algoritmos trabajados para la formación y barrido del mundo, y trabajar en un ambiente de dos y tres dimensiones. El algoritmo de formación utiliza una recompensa acumulada

del conjunto de agentes, y el algoritmo de barrido toma en cuenta los espacios que ya visito, el ambiente tri-dimensional toma en cuenta en que nivel se encuentra y con esa información penaliza al agente. Este algoritmo retorna los estados necesarios para ejecutar la tarea aprendida. Ambos entrenamientos emiten salidas en archivos de texto los cuales son utilizados para mostrar una visualización del funcionamiento del algoritmo en dos y tres dimensiones.

Capítulo 4

Resultados

Q-Learning es un algoritmo de aprendizaje por refuerzo el cual fue implementado para resolver el problema de formación y barrido de espacios en un mundo de cuadrícula dos-Dimensiones y en tres-Dimensiones. Al principio de este capítulo se explican los experimentos realizados en dos-Dimensiones de barrido de un espacio y formación de agentes para determinar los parámetros adecuados. Luego se presentan los resultados en tres-Dimensiones de barrido de un espacio y formación de agentes. Finalmente se presentan los resultados de animaciones para dos dimensiones y tres-dimensiones.

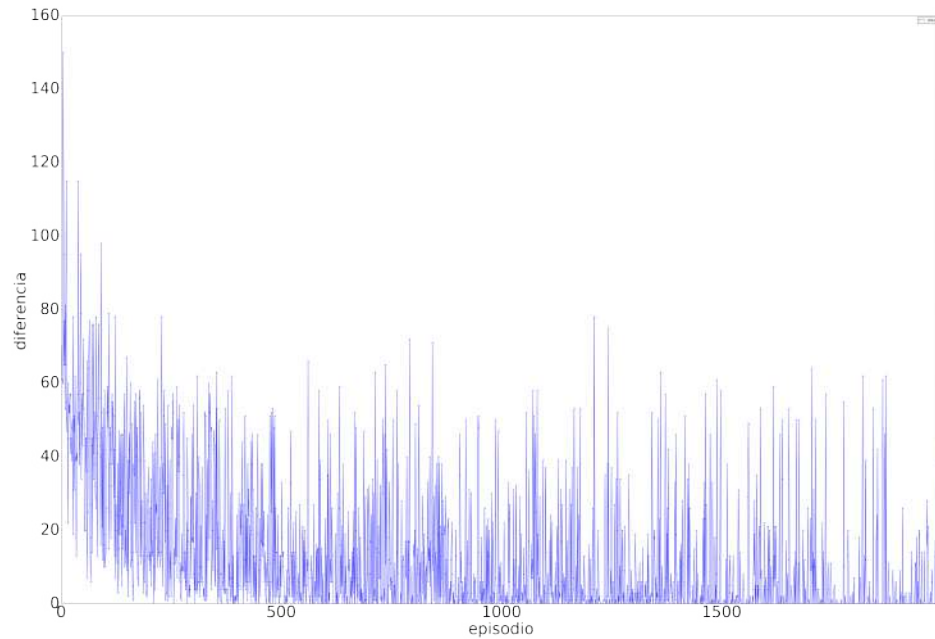
4.1. Escenarios experimentales en dos dimensiones

El problema consiste en que un conjunto de N agentes se organicen y cubran un espacio bi-dimensional de $n \times m$. Se probaron 5 formaciones distintas donde se realiza una variación en los parámetros mencionados. Dependiendo de la formación se entreno una reconfiguración de los agentes para poder ayudar con el desplazamiento en el mundo.

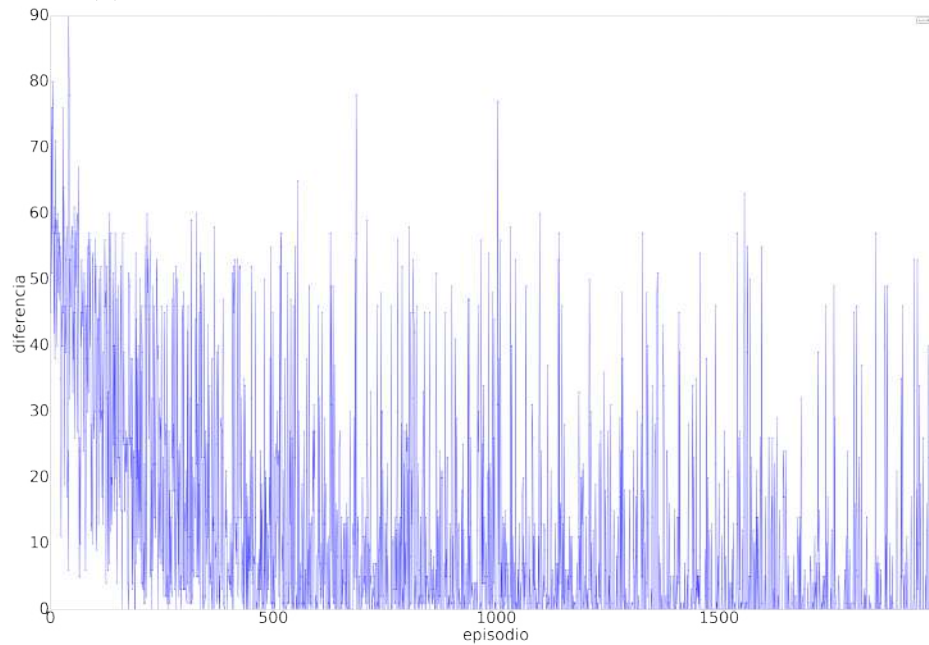
4.1.1. Experimento para barrido de espacio

El problema consiste en que el conjunto de agentes se mueva como uno solo en todo el espacio. Para determinar que parámetros mejor resuelve este problema se realizaron algunas pruebas donde se obtuvieron los siguientes resultados. Las dimensiones para todas las pruebas fue de 3×3 y 4×4

Experimentos de 3×3 :



(a) Recorrido A: $\alpha = ,4$, $\epsilon = ,22$, con promedio de 3 corridas



(b) Recorrido B $\alpha = ,5$, $\epsilon = ,1$, con promedio de 3 corridas

Figura 4.1: Gráficas comparativas de diferencia en Q por episodios con 100 pasos como máximo y 2000 episodios.

En la gráfica 4.1 podemos ver como una alpha de ,4, en la gráfica 4.1a obtiene mejores que usando una alpha de .5 como en la gráfica 4.1b.

Como prueba final al algoritmo despues de entrenar se le pide la solución desde un punto inicial en el mundo y retorna un camino completo en un archivo

txt como el que se muestra a continuación en texto y en la figura 4.2.

****GETTING PATH****

0100000000

Agent Pos = 0 Agent Action= 3 Agent New Pos1

1110000000

Agent Pos = 1 Agent Action= 3 Agent New Pos2

2111000000

Agent Pos = 2 Agent Action= 4 Agent New Pos5

5111001000

Agent Pos = 5 Agent Action= 4 Agent New Pos8

8111001001

Agent Pos = 8 Agent Action= 1 Agent New Pos7

7111001011

Agent Pos = 7 Agent Action= 0 Agent New Pos4

4111011011

Agent Pos = 4 Agent Action= 1 Agent New Pos3

3111111011

Agent Pos = 3 Agent Action= 4 Agent New Pos6

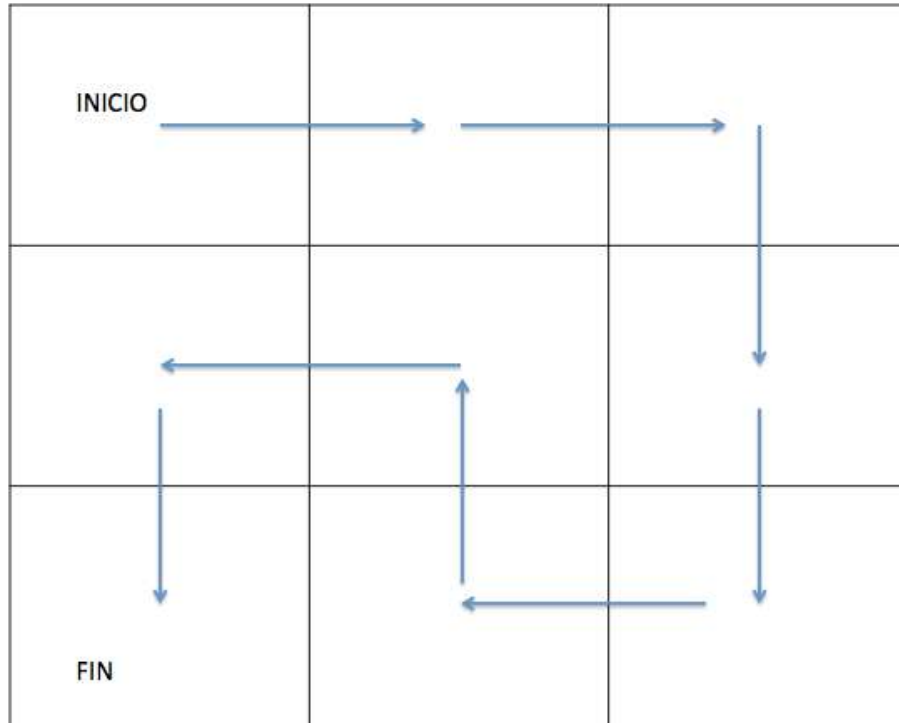
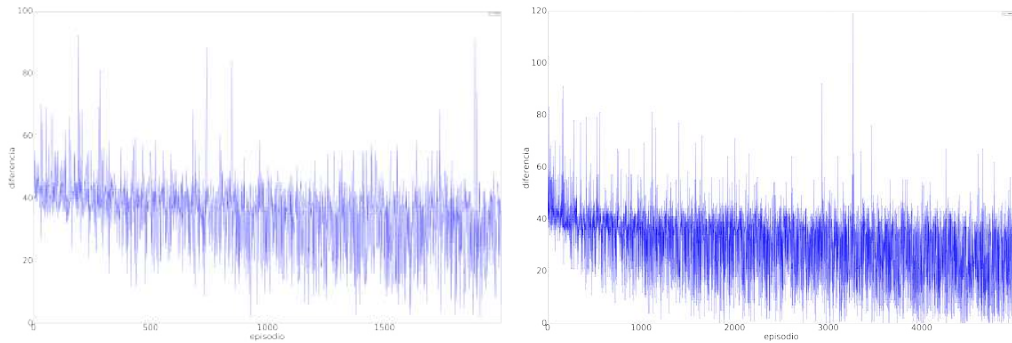
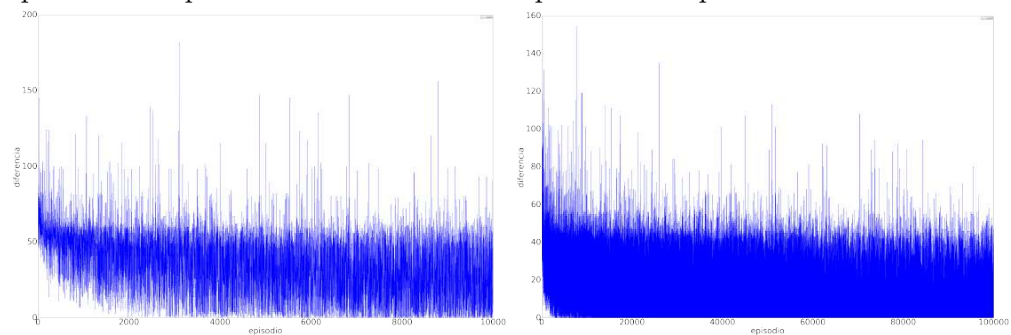


Figura 4.2: Ilustración que muestra el punto inicial del mundo con índice 0 y finaliza en el índice 6.

Experimentos de 4×4 :



(a) Recorrido A: $\alpha = ,4$, $\epsilon = ,1$, a 2000 episodios con promedio de 3 corridas (b) RecorridoB $\alpha = ,4$, $\epsilon = ,1$, a 5000 episodios con promedio de 3 corridas



(c) RecorridoB $\alpha = ,4$, $\epsilon = ,1$, a 10000 con promedio de 3 corridas (d) RecorridoB $\alpha = ,4$, $\epsilon = ,1$ a 100,000 episodios con promedio de 3 corridas

Figura 4.3: Gráficas comparativas de diferencia en Q por episodios con 100 pasos como máximo y 2000,5000, 10000 y 100000 episodios.

Como prueba final al algoritmo despues de entrenar se le pide la solución desde un punto inicial en el mundo y retorna un camino completo en un archivo txt como el que se muestra a continuación en texto y en la figura 4.4.

****GETTING PATH****

010000000000000000

Agent Pos = 0 Agent Action= 3 Agent New Pos1

111000000000000000

Agent Pos = 1 Agent Action= 4 Agent New Pos5

511000100000000000

Agent Pos = 5 Agent Action= 3 Agent New Pos6

611000110000000000

Agent Pos = 6 Agent Action= 0 Agent New Pos2

2111001100000000

Agent Pos = 2 Agent Action= 3 Agent New Pos3

3111101100000000

Agent Pos = 3 Agent Action= 4 Agent New Pos7

7111101110000000

Agent Pos = 7 Agent Action= 4 Agent New Pos11

111111011100010000

Agent Pos = 11 Agent Action= 4 Agent New Pos15

151111011100010001

Agent Pos = 15 Agent Action= 1 Agent New Pos14

141111011100010011

Agent Pos = 14 Agent Action= 0 Agent New Pos10

101111011100110011

Agent Pos = 10 Agent Action= 1 Agent New Pos9

91111011101110011

Agent Pos = 9 Agent Action= 4 Agent New Pos13

131111011101110111

Agent Pos = 13 Agent Action= 1 Agent New Pos12

121111011101111111

Agent Pos = 12 Agent Action= 0 Agent New Pos8

811110111111111111

Agent Pos = 8 Agent Action= 0 Agent New Pos4

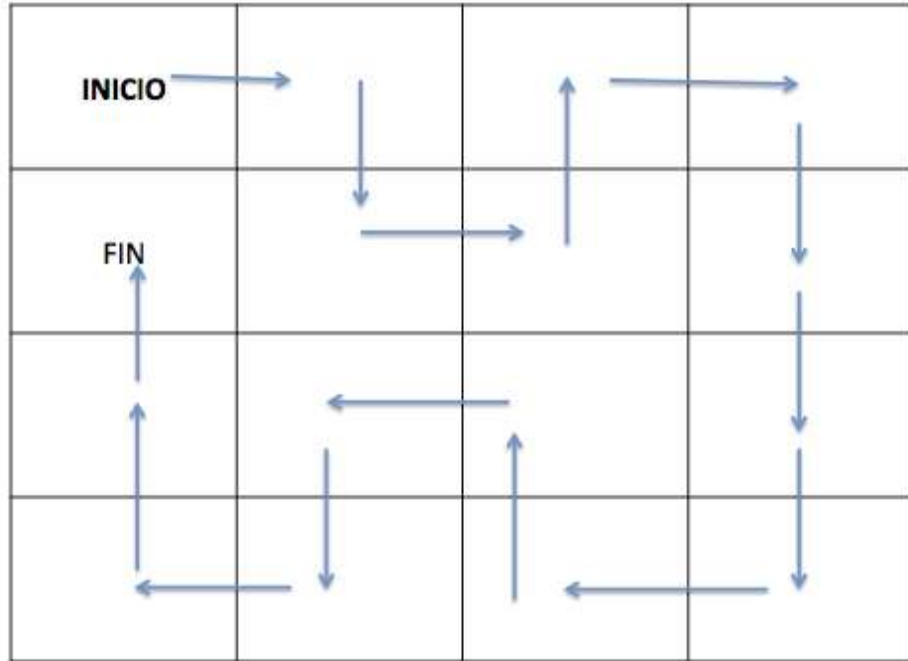


Figura 4.4: Ilustración que muestra el punto inicial del mundo con índice 0 y finaliza en el índice 6.

4.1.2. Experimento para formación de agentes

En este apartado se muestran los resultados de la solución para las diferentes formaciones a realizar por los agentes y pruebas de diferentes parámetros que mejoran la convergencia del algoritmo.

Primera Formación: En esta primera formación el objetivo es formar un triángulo con 3 agentes, ver figura 4.5.

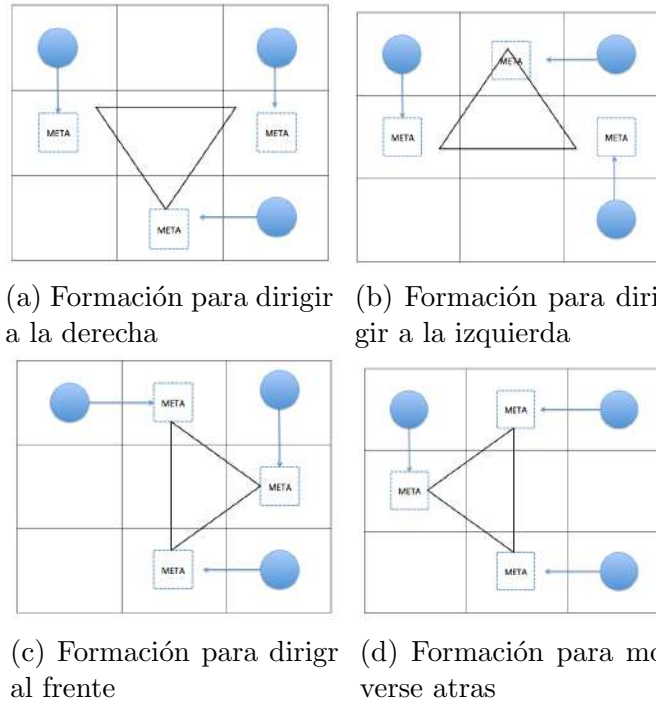
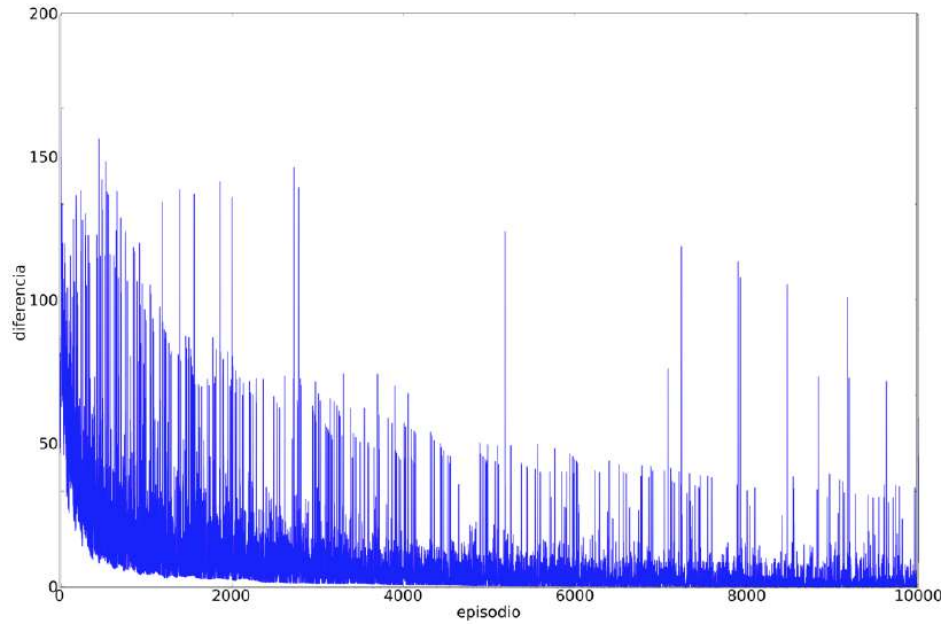


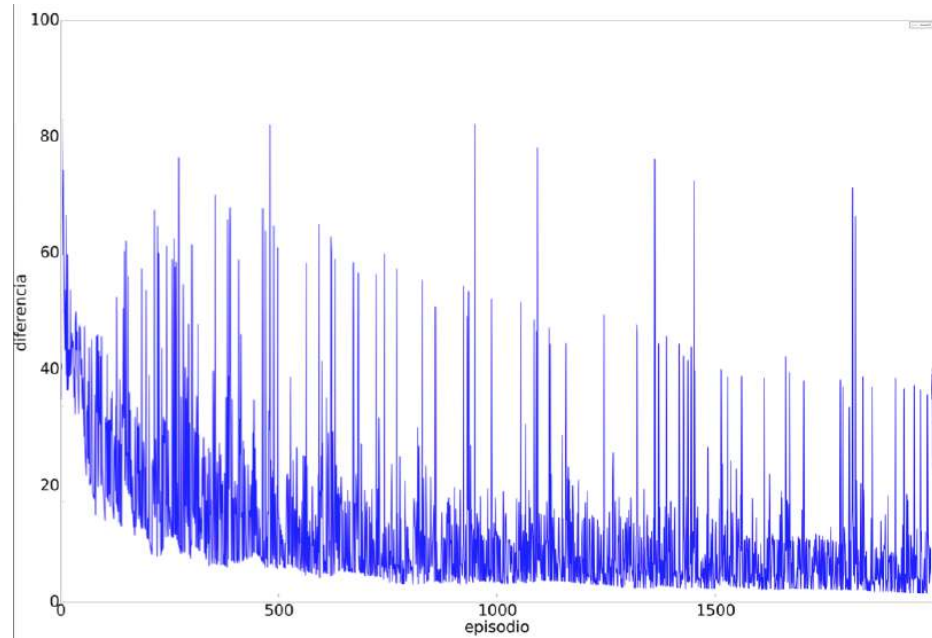
Figura 4.5: Metas de las diferentes direcciones para la formación de 3 agentes en triángulo

Para hallar los parámetros adecuados y entrenar cada formación se probaron diversos parámetros con una sola dirección. Esto es así por que las direcciones para esta formación son muy similares y no tiene sentido probar para cada dirección. A continuación se exponen los resultados encontrados.

Primer Experimento: Para esta primera prueba se inicio con los siguientes parámetros, y se obtuvo las siguientes gráfica 4.6



(a) Recorrido A: $\alpha = ,5$, $\epsilon = ,1$ a 10000 episodios con promedio de 3 corridas



(b) RecorridoB $\alpha = ,5$, $\epsilon = ,1$ a 2000 episodios con promedio de 3 corridas

Figura 4.6: Gráficas comparativas de diferencia en Q por episodios con 200 pasos como máximo y 2000 a 10000 episodios.

Para la formación de la figura 4.5 se puede observar que el experimento obtiene buenos resultados y con más iteraciones aprende una póliza optima

Segunda Formación: En esta segunda formación el objetivo es formar una cruz con 4 agentes, ver figura 4.7. En este caso los agentes solo ocupan esta formación a lo largo de todo el trayecto, por lo cual no es necesario entrenarlos de otra manera.

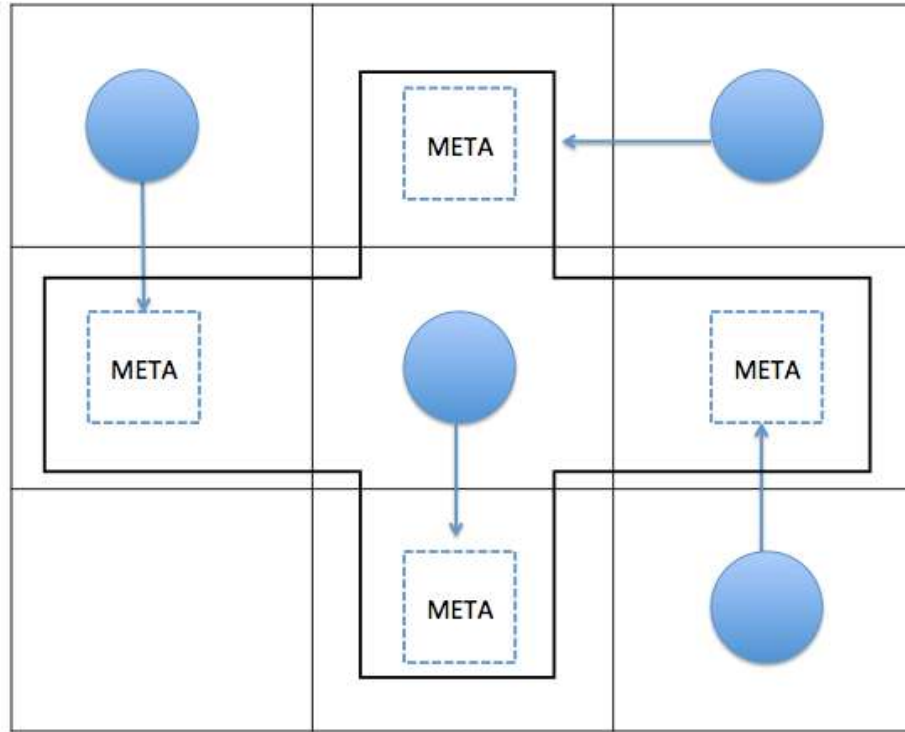
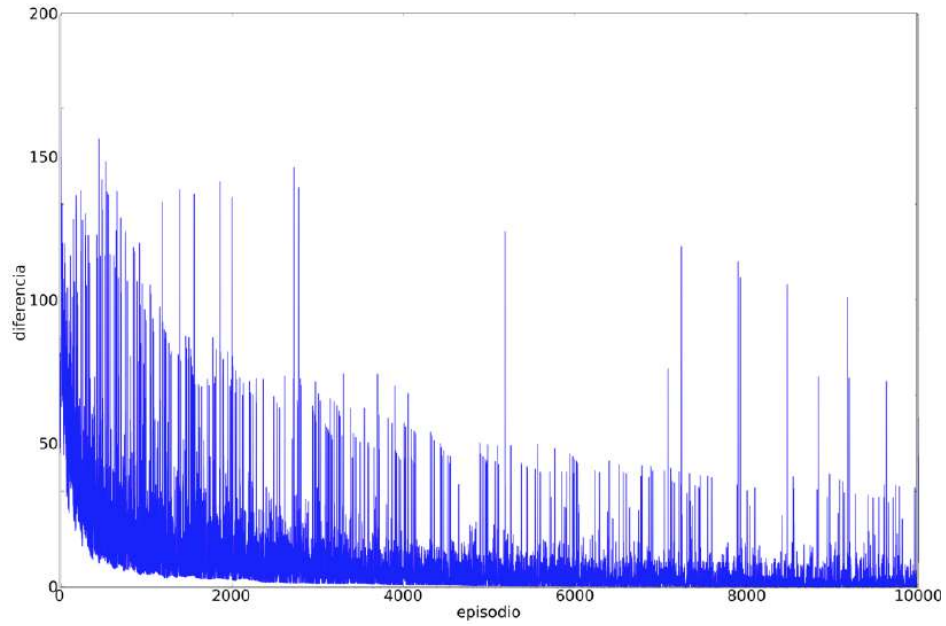
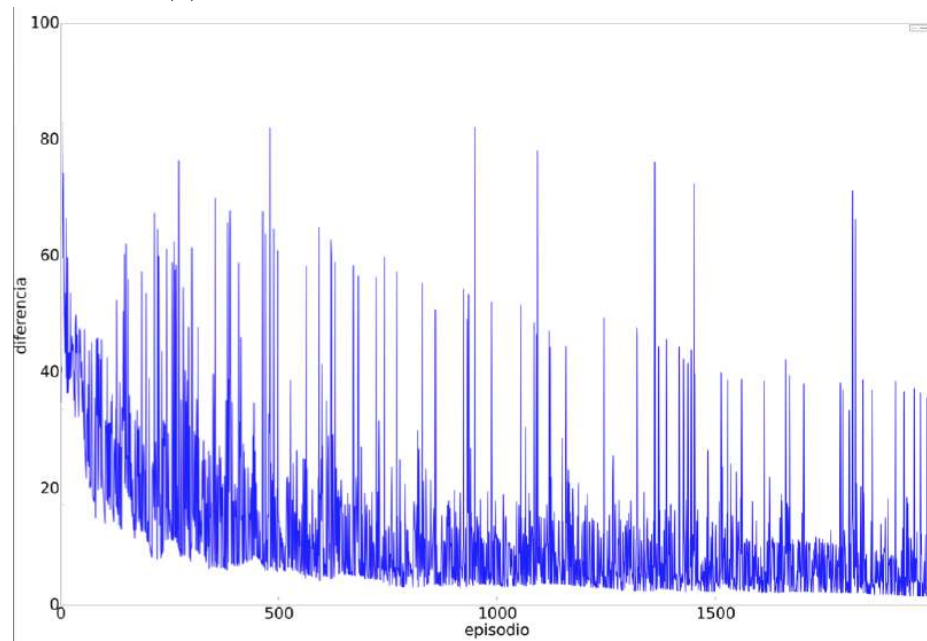


Figura 4.7: Formación de 4 agentes en cruz.

Experimento: En este experimento se utilizaron los siguientes parámetros:
En la gráfica 4.8 se tiene los resultados con los parámetros ya mencionados. Podemos ver que aún puede necesitar más iteraciones o ajustar la α .



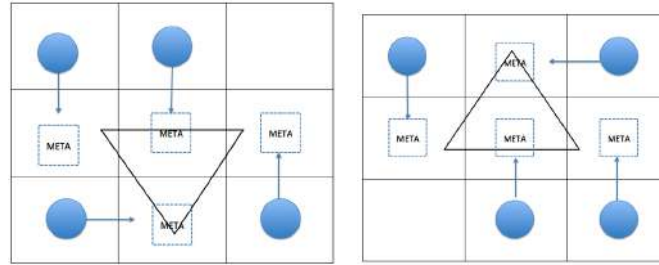
(a) Recorrido A: $\alpha = ,5$, $\epsilon = ,1$ a 10000 episodios



(b) RecorridoB $\alpha = ,5$, $\epsilon = ,1$ a 2000 episodios

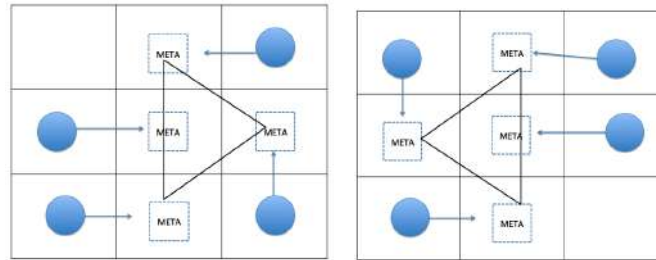
Figura 4.8: Gráficas comparativas de diferencia en Q por episodios con 200 pasos como máximo y 2000 a 10000 episodios.

Tercera Formación: En esta tercera formación el objetivo es formar un triángulo, similar a la primera formación, pero utilizando 4 agentes, ver figura 4.9.



(a) Formación para dirigir a la derecha

(b) Formación para dirigir a la izquierda

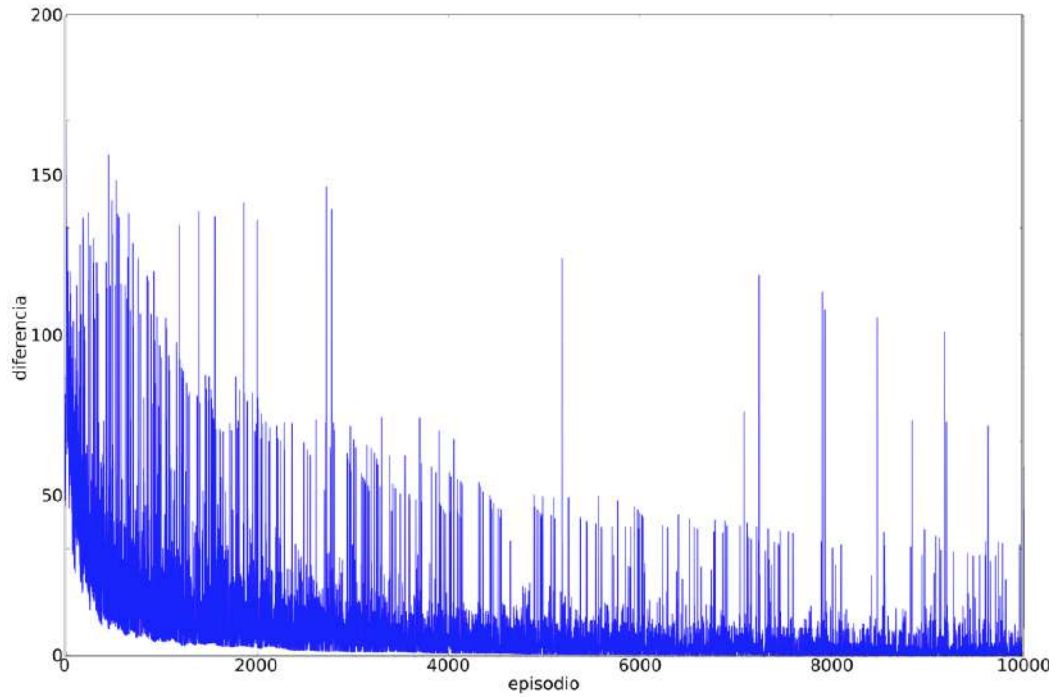


(c) Formación para dirigirse al frente

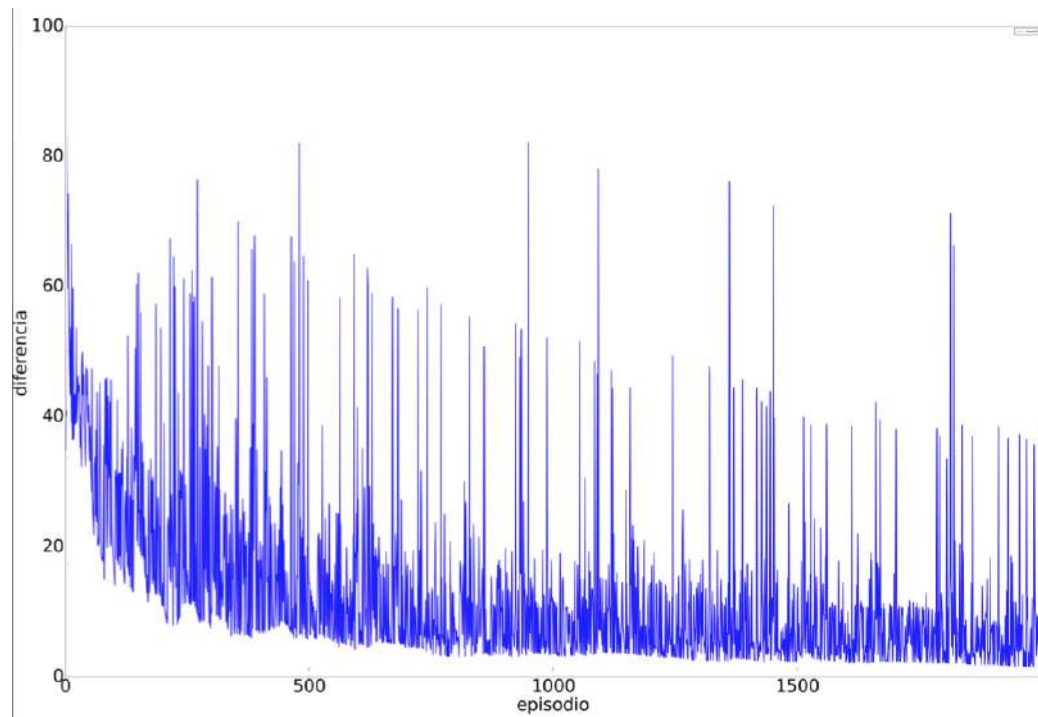
(d) Formación para moverse atrás

Figura 4.9: Metas de las diferentes direcciones para la formación de 4 agentes en triángulo

Experimento: En la gráfica 4.10 se tiene los resultados con los parámetros ya mencionados.



(a) Recorrido A: $\alpha = ,5$, $\epsilon = ,1$ a 10000 episodios

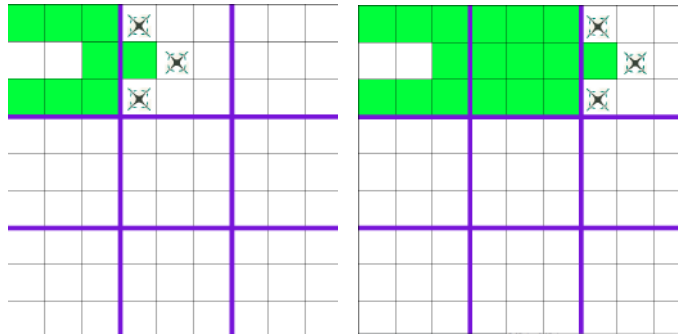


(b) Recorrido B $\alpha = ,5$, $\epsilon = ,1$ a 2000 episodios

Figura 4.10: Gráficas comparativas de diferencia en Q por episodios con 200 pasos como máximo y 2000 a 10000 episodios.

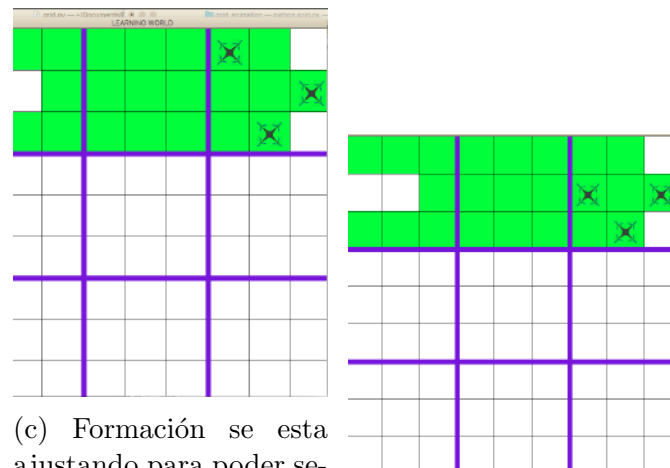
4.1.3. Animación en dos Dimesiones

En esta sección se muestran algunas imagenes de como se está utilizando el conocimiento para gráficar en un visualizador de dos dimensiones escrito en python para ver los movimientos de los agentes.



(a) Formación se mueve hacia el frente

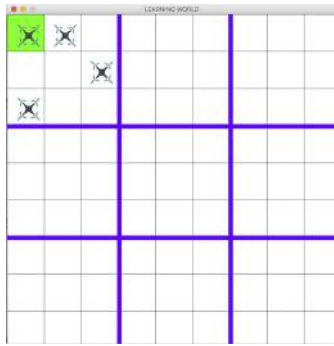
(b) Formación llega al final del mundo



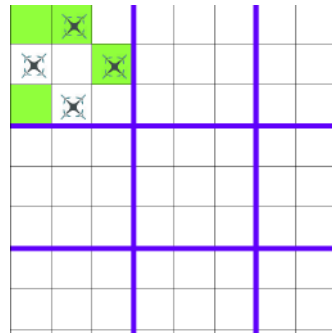
(c) Formación se esta ajustando para poder seguir el trayecto del conocimiento.

(d) Formación termina de ajustarse

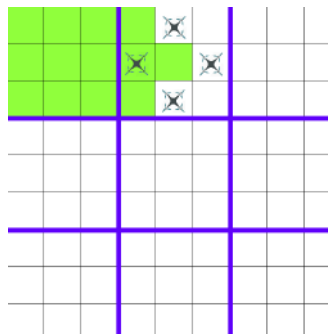
Figura 4.11: Simulación en 2 dimensiones utilizando el conocimiento de barrido y de formación en triángulo de 3 agentes



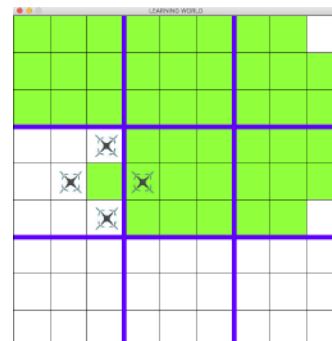
(a) Formación se mueve hacia el frente



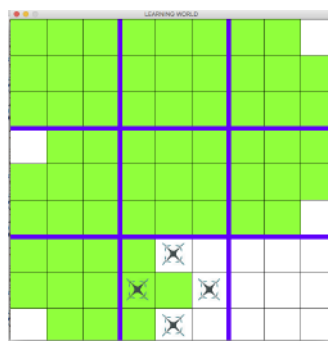
(b) Formación llega al final del mundo



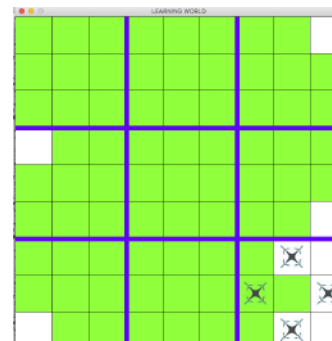
(c) Formación se está ajustando para poder seguir el trayecto del conocimiento.



(d) Formación termina de ajustarse

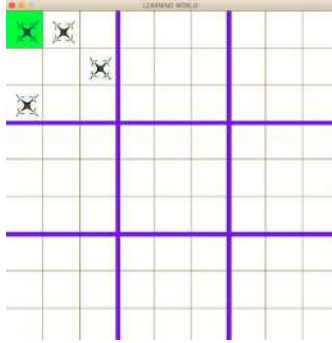


(e) Formación se está ajustando para poder seguir el trayecto del conocimiento.

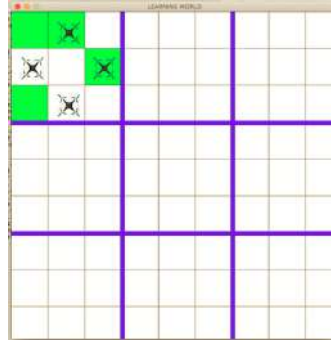


(f) Formación termina de ajustarse

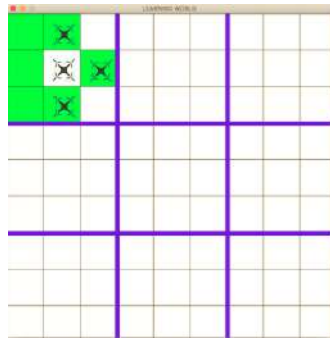
Figura 4.12: Simulación en 2 dimensiones utilizando el conocimiento de barrido y de formación en triángulo de 4 agentes en cruz



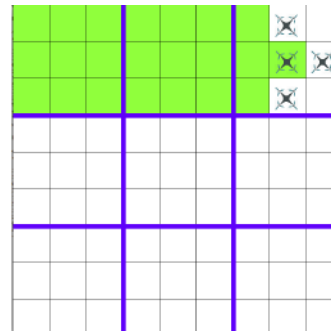
(a) Formación se mueve hacia el frente



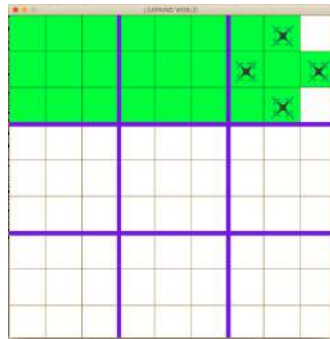
(b) Formación llega al final del mundo



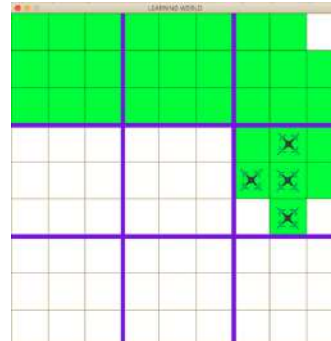
(c) Formación se esta ajustando para poder seguir el trayecto del conocimiento.



(d) Formación termina de ajustarse



(e) Formación se esta ajustando para poder seguir el trayecto del conocimiento.



(f) Formación termina de ajustarse

Figura 4.13: Simulación en 2 dimensiones utilizando el conocimiento de barrido y de formación en triangulo de 4 agentes en triangulo

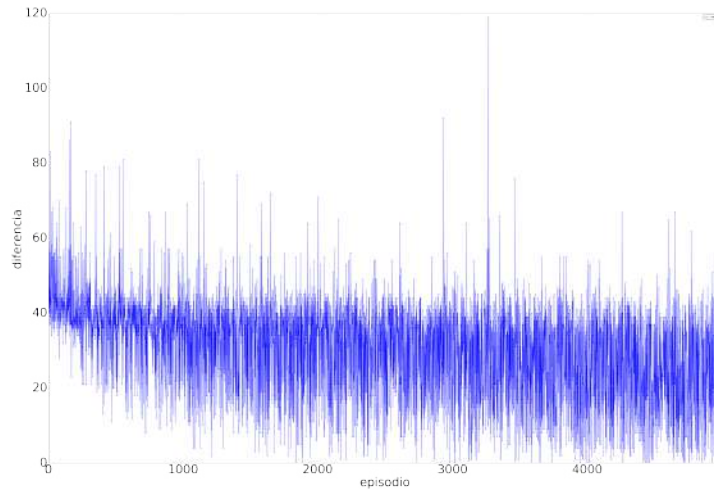
4.2. Escenarios experimentales en tres dimensiones

En esta sección se exponen los resultados para el ambiente de tres dimensiones,

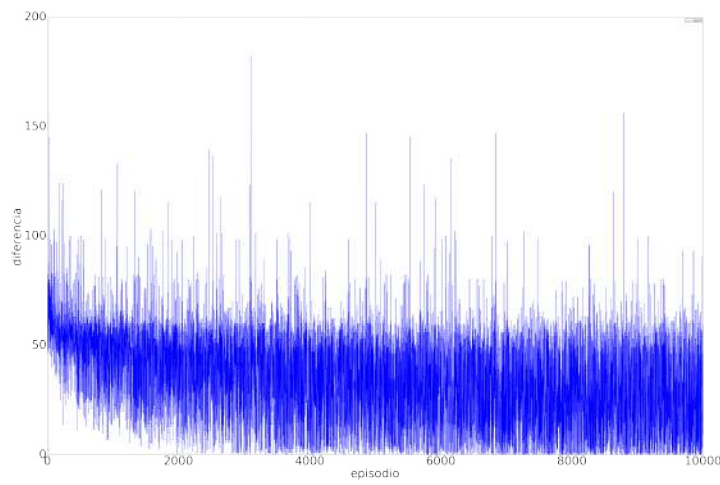
4.2.1. Experimento para barrido de espacio

El problema consiste en que el conjunto de agentes se mueva como uno solo en todo el espacio. Para determinar que parámetros mejor resuelve este problema se realizaron algunas pruebas donde se obtuvieron los siguientes resultados. Las dimensiones para todas las pruebas fue de 3×3

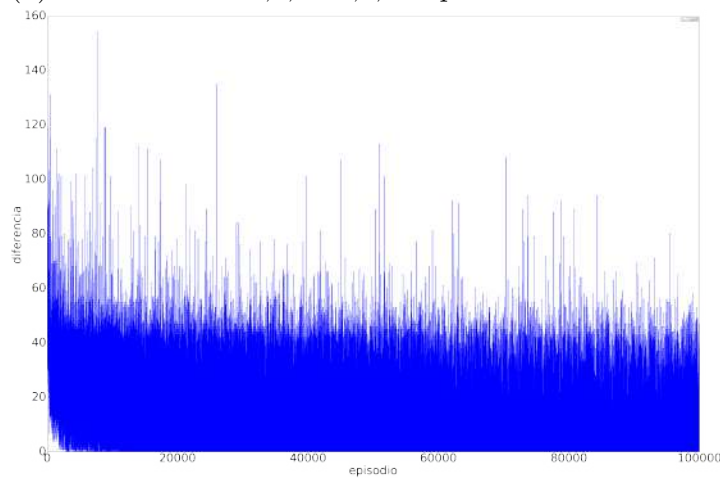
Experimentos de $3 \times 3 \times 3$:



(a) Recorrido A: $\alpha = ,4$, $\epsilon = ,22$, con promedio de 3 corridas



(b) Recorrido B $\alpha = ,5$, $\epsilon = ,1$, con promedio de 3 corridas



(c) Recorrido B $\alpha = ,5$, $\epsilon = ,1$, con promedio de 3 corridas

Figura 4.14: Gráficas comparativas de diferencia en Q por episodios con 100 pasos como máximo y 2000 episodios.

Como prueba final al algoritmo despues de entrenar se le pide la solución desde un punto inicial en el mundo y retorna un camino completo en un archivo txt como el que se muestra a continuación en texto

****GETTING PATH****

```
0 100000000 000000000 000000000 0 UP
0 100000000 100000000 000000000 1 up
0 100000000 100000000 100000000 2 RIGHT
3 100000000 100000000 100100000 2 RIGHT
6 100000000 100100100 100100100 2 FORWARD
7 100000000 100100110 100100110 2 FORWARD
8 100000000 100100111 100100111 2 LEFT
5 100000000 100101111 100101111 2 BACKWARDS
4 100000000 100111111 100111111 2 LEFT
1 100000000 110111111 110111111 2 FORWARD
2 100000000 110111111 111111111 2 STAY
```

4.2.2. Animación en tres Dimensiones

En esta sección se muestran algunas imágenes de como se está utilizando el conocimiento para graficar en un visualizador de 2-dimensiones escrito en python para ver los movimientos de los agentes.

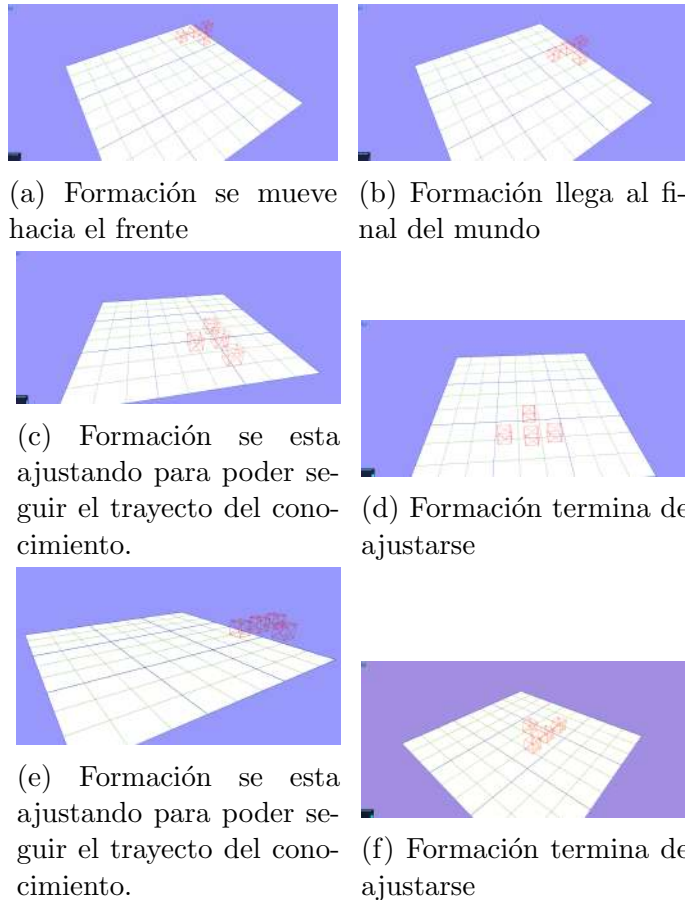


Figura 4.15: Simulación en 2 dimensiones utilizando el conocimiento de barrido y de formación en triangulo de 4 agentes en triangulo

4.2.3. Resumen

En este capítulo se utilizó el algoritmo de 2 dimensiones de q-learning para recorrer un grid de 3×3 se probaron distintos parámetros para lograr obtener un aprendizaje eficiente. A demas se probaron diferentes formaciones en un grid de 3×3 para ver sus rendimientos de aprendizaje.

Por otro lado se extendio el problema de cubrir un espacio discreto a tres dimensiones, dando resultados satisfactorios, ya que a pesar de tener un número alto de iteraciones lograr encontrar un caminno.

Estos aprendizajes fueron implementados en simuladores de dos dimensiones y de tres dimensiones con sus respectivas formaciones.

Capítulo 5

Conclusiones

En el primer caso para un mundo de dos dimensiones para realizar su recorrido total, los mejores valores son aquellos que tienen un valor de alpha medio alto y un valor de ϵ de 0.22 . Concretamente, hemos decidido escoger los valores de $\alpha = 0,4$ y $\epsilon = 0.22$; que utilizaremos en el resto de nuestros experimentos. Esto se debe a que, como puede observarse en las distintas gráficas, son los valores que ofrecen una mejora en el aprendizaje, pero a la vez más fiable y se va disminuyendo el error a largo plazo.

Por otro lado podemos afirmar, que un valor de alpha un poco más alto no hacen que el agente aprenda más rápido, es más lento su aprendizaje.

Esto se debe a que estos resultados han sido obtenidos siguiendo una estrategia de recompensa conforme a sus acciones (que sólo asigna recompensa en la meta), y una política de selección de acciones probabilística. Si modificáramos tanto una como la otra, los resultados obtenidos serían distintos.

Analizando los experimentos para un barrido de un mundo de tres dimensiones se obtuvieron tres gráficas para realizar una comparación. Para este mundo se aplicaron los mismos valores para la sesión de experimentos. Donde se concluye que los mismos parametros para el mundo funcionan pero necesita un número mayor de iteraciones. Esto se debe a que el algoritmo necesita más tiempo para poder cubrir un número mayor de estados para poder encontrar la solución.

Ahora para el caso de las formaciones se esta utilizando las alphas ya encontrada como la mejor. Como muchas de las formaciones son parecidas solo fue necesario hacer experimentos con una dirección de la formación ya que las de más solo sería variar los puntos inicio y meta. Como en el experimento del barrido se calculó un error de cambio en Q por episodio. Se encontro que mientras más episodios se da el error seguira disminuyendo, pero se dio que con 4000 episodios o más es suficiente para aprender diferentes formación y direcciones para el grupo de agentes.

Bibliografía

- [1]
- [2] A. Abraham, H. Guo, and H. Liu. *Swarm intelligence: foundations, perspectives and applications*. Springer, 2006.
- [3] B. Bachir, A. Ali, and M. Abdellah. Multiobjective Optimization of an Operational Amplifier by the Ant Colony Optimisation Algorithm. *Electrical and Electronic Engineering*, 2(4):230–235, 2012.
- [4] B. Bachir, A. Ali, and M. Abdellah. Multiobjective optimization of an operational amplifier by the ant colony optimisation algorithm. *Electrical and Electronic Engineering*, 2(4):230–235, 2012.
- [5] J. C. Barca, A. Sekercioglu, and A. Ford. *Controlling Formations of Robots with Graph Theory*, pages 563–574. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [6] M. A. Batalin, G. S. Sukhatme, and M. Hattig. Mobile robot navigation using a sensor network. *Proc. IEEE International Conference on Robotics and Automation ICRA '04*, 1:636–641, 2004.
- [7] S. Bhattacharya, M. Likhachev, and V. Kumar. Distributed path consensus algorithm. *University of Pennsylvania*, pages 1–37, 2010.
- [8] V. U. Cetina. Aprendizaje por refuerzo. 2012.
- [9] D. W. Corne, A. Reynolds, and E. Bonabeau. Swarm intelligence. In *Handbook of Natural Computing*. 1599-1622, Springer, 2012.
- [10] M. Dervisi. Terrain coverage ant algorithms: The random kick effect. 2013.
- [11] R. C. Eberhart and Y. Shi. Particle swarm optimization: developments, applications and resources. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 81–86. IEEE, 2001.

- [12] S. Hert, S. Tiwari, and V. Lumelsky. A terrain-covering algorithm for an auv. *Autonomous Robots*, 3(2):91–119, Jun 1996.
- [13] N. R. Hoff III, A. Sagoff, R. J. Wood, and R. Nagpal. Two foraging algorithms for robot swarms using only local communication. In *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*. 123-130, IEEE, 2010.
- [14] M. Hutson. Self-taught artificial intelligence beats doctors at predicting heart attacks, 2017.
- [15] A. Jevtić and D. Andina de la Fuente. Swarm intelligence and its applications in swarm robotics. In *Proc . 6th WSEAS Int. Conference on Computational Intelligence, Man-Machine Systems and Cybernetics*. 41-46, WSEAS, 2007.
- [16] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, pages 237–285, 1996.
- [17] S. Keerthi, K. Ashwini, and M. Vijaykumar. Survey paper on swarm intelligence. *International Journal of Computer Applications*, 8-12, 115(5), 2015.
- [18] M. T. Labs. How businesses can leverage reinforcement learning?
- [19] B. Marr. The top 10 ai and machine learning use cases everyone should know about, 2016.
- [20] D. Mellinger, M. Shomin, N. Michael, and V. Kumar. Distributed Autonomous Robotic Systems. *Springer Tracts in Advanced Robotics*, 83(August):545–558, 2013.
- [21] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.
- [22] J. Nimeroff. How machine learning will be used for marketing in 2017, 2017.
- [23] R. Poli. Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 1-10, 2008:3, 2008.
- [24] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.
- [25] S. Raschka. *Python Machine Learning*. Packt Publishing, 2015.

- [26] R. S. Sutton and A. G. Barto. Reinforcement Learning : An Introduction. 1998.
- [27] N. Tsourveloudis, L. Doitsidis, and K. Valavanis. Autonomous navigation of unmanned vehicles: A fuzzy logic perspective,” in cutting edge robotics, 07 2005.
- [28] C. M. Vigorito. Distributed path planning for mobile robots using a swarm of interacting reinforcement learners. *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems - AAMAS '07*, 5:1, 2007.
- [29] Z. Wang, L. Qin, and W. Yang. A self-organising cooperative hunting by robotic swarm based on particle swarm optimisation localisation. *International Journal of Bio-Inspired Computation*, 7(1):68–73, 2015.
- [30] S. Zaman, W. Slany, and G. Steinbauer. ROS-based Mapping , Localization and Autonomous Navigation using a Pioneer 3-DX Robot and their Relevant Issues. pages 0–4, 2011.