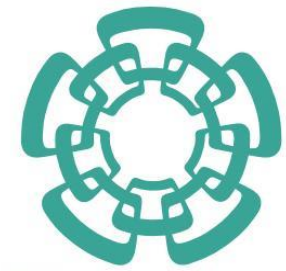

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE CONTROL AUTOMÁTICO

“Planeación de trayectoria en entornos desconocidos para sistemas multi-agentes utilizando aprendizaje por reforzamiento inteligente”

T E S I S

Que presenta:

David Luviano Cruz

Para obtener el grado de

DOCTOR EN CIENCIAS

EN LA ESPECIALIDAD DE

CONTROL AUTOMATICO

Director de la Tesis:

Dr. Wen Yu Liu

Ciudad de México **SEPTIEMBRE, 2016**

AGRADECIMIENTO

Antes de todo quiero agradecer a Dios la oportunidad de terminar mis estudios doctorales, también a mi esposa Diana y a mis padres Norma y Gerardo por el inmenso apoyo brindado, en especial a mi abuela Luz Maria por abrirme las puertas de su casa desde el primer momento que llegue la Ciudad de México, y demás familiares que siempre me apoyaron.

Toda mi gratitud hacia el Dr. Wen Yu Liu por su valiosa guía durante la realización de esta tesis doctoral, al igual que al CINVESTAV y todo el personal que lo conformo por siempre estar dispuestos a apoyarme, por ultimo mi más sentido agradecimiento al CONACYT y al gobierno de la republica ya que sin su apoyo económico hubiese sido imposible seguir mis estudios de posgrado.

Índice general

Índice de figuras

Capítulo 1

Introducción

Las investigaciones relacionadas con los sistemas multi-agentes (MAS) es un subcampo emergente de la inteligencia artificial distribuida, el cual tiene como objetivo proporcionar principios de construcción de sistemas complejos por medio de múltiples agentes y mecanismos para la coordinación de los comportamientos independientes de cada agente.

La razón mas importante para utilizar sistemas de multi-agentes es tener un modelo mas natural de las situaciones de la vida real que requieren de la cooperación de diferentes entidades. En particular existen personas con distintas perspectivas u organizaciones con diferentes objetivos, por lo que un sistema de multi-agentes es necesario para manejar dichas interacciones [1][82]. Sistemas multi-agentes son reconocidos como cruciales para muchos problemas en el mundo real.

Existen significadas características en los sistemas multi-agentes que lo distinguen de un sistema de control de un solo agente. Primero los agentes son considerados parcialmente autónomos esto es debido a que los agentes no pueden conocer toda la información global disponible y por ende solo pueden accesar a una limitada información, segundo los MAS son sistemas muy complejos,por consiguiente un agente individual no puede decidir una acción óptima solo usando sus conocimientos locales [83].

Por lo tanto, limitaciones en el poder de procesamiento de un solo agente son eliminados en un entorno de multi-agentes, ya que la información y el sistema de control son distribuidos,

los sistemas multi-agentes incluyen las ventajas que proporcionan los sistemas distribuidos tales como escalabilidad, tolerancia al error, paralelismo aplicado a problemas que debido a su naturaleza solo pueden ser atacados con múltiples agentes que observan y actúan desde diferentes lugares simultáneamente [23].

1.1. Motivación

Los sistemas multi-agentes han cobrado una importante relevancia en el estudio de problemas donde un enfoque centralizado no es el mas adecuado para describirlo, los agentes proporcionan una manera distinta de abordar y enfocar una problemática determinada, ya sea realizando una tarea completamente competitiva (cada agente tiene objetivos opuestos), tareas completamente cooperativas (donde los agentes desarrollan un objetivo común o tareas mixtas .

Los agentes al ser entidades autónomas que no dependen de un control centralizado, tienen la necesidad de generar un aprendizaje confiable de las acciones óptimas a realizar, con el objetivo de completar la tarea encomendada a ellos. El aprendizaje por reforzamiento es un campo ampliamente estudiado para el caso de sistemas donde un solo agente interactúa con el medio, diversas pruebas de convergencia han sido dadas a conocer y la efectividad de los algoritmos han sido probadas en diferentes problemáticas.

Sin embargo, en los sistemas multi-agentes no existe una correlación entre las acciones realizadas por cada uno de los agentes, ya que estos pueden perseguir objetivos finales distintos o incluso al buscar un objetivos comunes, dando pie a la necesidad de coordinación entre los agentes, por lo anterior, las bases teóricas en los que es asentada el aprendizaje por reforzamiento de un solo agente no se mantiene valida para sistemas multi-agentes.

En adición, el problema de la dimensionalidad en el aprendizaje por reforzamiento esta presente, es decir, el numero de pares estado-acciones ha aprender se incrementa de manera exponencial conforme el espacio de estados, el espacio de acciones y el numero de agentes presentes en el entorno aumentan. Por lo que la mayoría de los algoritmos de aprendizaje por reforzamiento son usados en sistemas donde las variables de estado son discretas.

En la presente tesis se proponen abordar técnicas para mitigar el problema de la dimensionalidad, por medio de una aproximación difusa de los estados presentes en el entorno, permitiendo el uso de variables de estado continuas durante el aprendizaje por reforzamiento, combinando el algoritmo clásico de Q-iteracion con un aproximador difuso, la base de reglas difusas recibe los estados como entrada y produce los Q-valores de las acciones discretas como salida. El algoritmo propuesto esta basado en un modelo del sistema en forma de función de transición de estados y función de recompensa.

La aproximación difusa fue escogida en virtud de que las funciones de membresias pueden ser vistas como funciones bases dependientes, al utilizar en esta propuesta funciones de membresias del tipo triangular, el análisis matemático se reduce ya que la proyección por mínimos cuadrados se reduce a una asignación en el vector de aproximación.

Con la intención de atacar el problema del consumo de recursos computacionales, en la propuesta es usada una red neuronal para aproximar las acciones a ejecutar por los agentes. Cuando estos se encuentren en estados desconocidos, es decir, en estados donde no se tengan disponible los Q-valores estado-accion óptimas, la red neuronal ofrece las acciones aproximadas sin la necesidad de volver a ejecutar el algoritmo de aprendizaje.

En el presente trabajo, se escogió utilizar una red neuronal en virtud de su capacidad de aproximación no lineal, donde los estados de los agentes son las entradas a la red neuronal y la salida son las acciones a ejecutar durante el entrenamiento supervisado de la misma.

El campo del aprendizaje por reforzamiento en sistemas multi-agentes ofrece grandes oportunidades de nuevos estudios, haciéndolo atractivo a distintas áreas donde podrían ser utilizadas como generación de trayectorias, minería de datos, comercio electrónico entre muchos otros.

1.2. Objetivos

Los objetivos principales buscados al realizar la presente tesis son:

1. Desarrollar una propuesta para mitigar el problema de la dimensionalidad en los problemas de aprendizaje por reforzamiento para sistemas multi-agentes por medio de una

aproximación difusa del espacio de estados, permitiendo el análisis de sistemas con espacio de estados continuos,además de generar una disminución de los pares estado-acción ha ser aprendidos.

2. Proponer el uso de una red neuronal para estimar las acciones óptimas a realizar por los agentes cuando estos se encuentren presentes en estados que no han sido visitados previamente y por lo tanto no han sido aprendidos, evitando volver a ejecutar el algoritmo y por lo consiguiente ahorrando tiempo y poder computacional

3. Validar las propuestas anteriores en tareas completamente cooperativas, mediante simulación usando el software Matlab .

4. Realizar experimentos donde se verifique la factibilidad de las técnicas propuestas.

5. Analizar los resultados obtenidos,además de exponer las conclusiones y alcances de las propuestas planteadas.

1.3. Estructura de la tesis

El capitulo 1 ofrece las motivaciones y objetivos concretos buscados en la presente tesis, así como las publicaciones obtenidas durante el periodo de estudio doctoral. El capitulo 2 ofrece una perspectiva general del estado de los sistemas multi-agentes presentes en las principales áreas donde estos son utilizados, además se presentan los fundamentos teóricos en las cuales se basa el aprendizaje por reforzamiento, se define un proceso de Markov para el caso de un solo agente y es dada la definición de un juego estocástico para sistemas multi-agentes, los objetivos de aprendizaje para las distintas tareas son formalizadas.

En el capitulo 3 se introduce una red neuronal con la finalidad de aproximar las acciones óptimas para cada agente presente, esta red neuronal es entrenada por medio de los datos obtenidos de un algoritmo de aprendizaje por reforzamiento clásico, donde la entrada son los estados del sistema y la salida son las acciones a tomar por los agentes, se presenta simulaciones así como casos experimentales

En el capítulo 4 una aproximación difusa de las variables de estado del sistemas es propuesta, dando la posibilidad de extender el análisis a sistemas con espacio de estados continuos, para lo anterior se utiliza un mapeo poyectivo para asegurar la convergencia del algoritmo, donde se demuestra que el algoritmo propuesto es una contracción. También se presentan simulaciones para validar la técnica propuesta.

En el ultimo capitulo 5 las conclusiones obtenidas durante el desarrollo de la tesis son expuestas, así como los trabajos futuros a desarrollar.

1.4. Publicaciones Obtenidas

Revista

1. David Luviano Cruz, Wen Yu, Path Planning of Multi-Agent Systems in Unknown Environment with Kernel Smoothing and Reinforcement Learning, *Neurocomputing,* Aceptado

2. David Luviano Cruz, Wen Yu, Multi agent reinforcement learning with approximate fuzzy iteration, *International Journal of Control, Automation and Systems*, sometido

Congreso

1. David Luviano Cruz, Wen Yu, Multi-Agent Path Planning in Unknown Environment with Reinforcement Learning and Neural Network, *IEEE International Conference on Systems, Man, and Cybernetics (SMC14)*, San Diego, USA, 3469-3474, 2014

2. David Luviano and Wen Yu, Path Planning in Unknown Environment with Kernel Smoothing and Reinforcement Learning for Multi-Agent Systems, *12th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE'15)*, Mexico City, Mexico, 2015

Capítulo 2

Multi-agentes y aprendizaje por reforzamiento

Un sistema multi-agente es un grupo de entidades autónomas que interactúan y comparten un entorno común el cual los agentes perciben por medio de sensores y lo modifican por medio de actuadores, estos agentes coexisten e interactúan en diferentes formas con otros agentes con el objetivo de optimizar alguna medida de desempeño [84],ejemplo de lo anterior pueden incluir seres humanos (tienen ojos como sensores y manos como actuadores),agentes robóticos (cámaras como sensor y ruedas como actuador),agentes de software(interfaces gráficas para el usuario como sensores) .

Los sistemas multi-agentes han encontrado aplicaciones en una gran variedad de campos tales como equipos de robots jugando fútbol soccer como lo muestra la Figura 2.1 [85], control distribuido,vehículos aéreos no tripulados [86], control de formación [87], administración de recursos y trafico , soporte de sistemas , minería de datos,ingeniería de diseño, búsqueda inteligente, diagnósticos médicos,entrega de productos, entre otros.

Una de sus ventajas de los sistemas multi-agentes es que pueden surgir como la manera mas natural de observar un sistema o pueden proveer de perspectiva alterna a sistemas que son considerados inicialmente como centralizados, una característica interesante de los MAS es que cada agente puede tomar decisiones de manera individual en base de su percepción

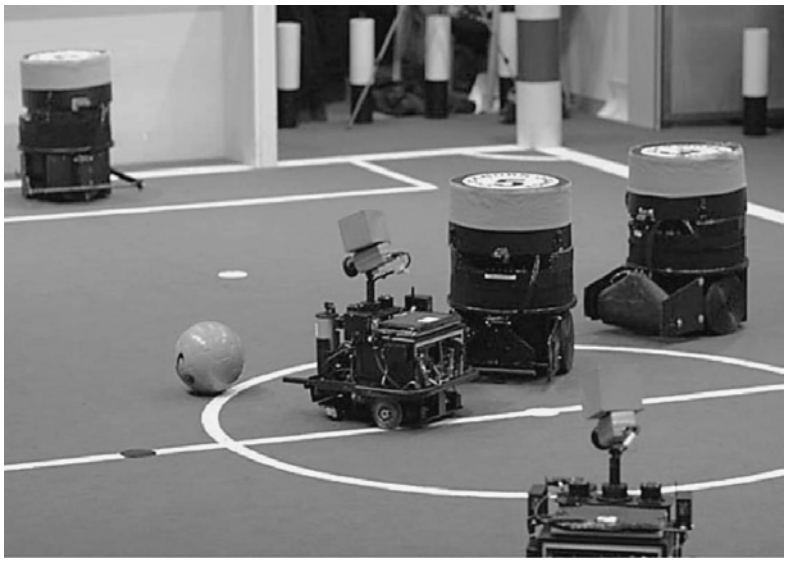

Figura 2.1: Ejemplo de sistema multi agente

local del medio tal como ha sido observado en diversos hechos biológicos.

Aplicaciones que implican o requieren múltiples agentes se están volviendo comunes en el mismo grado como lo hacen los robots y los agentes de software en la vida cotidiana. Un agente es la combinación de 3 componentes :percepción ,razonamiento y acción. Un agente recibe observaciones sobre el estado del medio ambiente y escoge una acción disponible para modificarlo, la parte de razonamiento es el responsable de procesar las percepciones recibidas para elegir una acción. Los agentes muy comúnmente tienen un objetivo, este puede ser alcanzar un estado deseado del medio o maximizar alguna señal individual o colectiva.

2.1. Características de los sistemas multi-agentes

Es común que varios agentes que componen un MAS sean diseñados de diferentes maneras. Los distintos diseños pueden involucrar variaciones en el hardware por ejemplo robots que juegan fútbol soccer basados en diferentes plataformas mecánicas, o variaciones en el

software donde agentes ejecutan diferentes tipos de código. Los agentes que están basados en diferentes tipo hardware o implementan diferentes conductas son llamados heterogéneos, en contraste con los agentes homogéneos que son diseñados de una manera idéntica y tienen las mismas capacidades a priori [26]

Los agentes tienen que lidiar con ambientes que pueden ser estáticos o dinámicos (cambian con el tiempo), deterministicos(una acción tiene un único efecto) o no determinista, discretos(existe un numero finito de acciones y percepciones en el) o continuos , accesibles(el agente obtiene de manera completa y precisa información sobre el estado del medio) o inaccesibles (mundo real) [45]. Por ejemplo la mayoría de las técnicas existentes de inteligencia artificial para agentes individuales han sido desarrollados en medios estáticos ya que son mas fáciles de manejar y permite un riguroso tratamiento matemático. En MAS,con la mera presencia de múltiples agentes hacen parecer a un entorno estático como dinámico desde el punto de vista de otros agentes.

La información colectiva que es obtenida por medio de los sensores de los agentes en un MAS es distribuida : los agentes pueden observar datos que difieren de manera espacial (en posiciones diferentes) , temporalmente (en diferentes tiempos) o de manera semántica (requiere diferentes interpretaciones). El hecho que los agentes puedan observar distintas cosas hace al medio parcialmente observable para cada agente , el cual tiene consecuencias en el proceso de toma de decisión de cada uno [88].

Contrariamente a los sistemas de un solo agente, el sistema de control en un MAS es típicamente descentralizado, esto quiere decir que el proceso de decisión de cada agente se encuentra en gran medida en el propio agente, el control descentralizado es preferido sobre el control centralizado por razones de robustez y de tolerancia a errores .

En un sistema de un solo agente se asume típicamente que el agente conoce sus propias acciones disponibles pero no necesariamente como el medio es afectado por sus ellas. En un sistema multi-agente los niveles de conocimiento de cada agente acerca el estado actual del medio pueden diferir sustancialmente. En un grupo de MAS que envuelven dos agentes homogéneos , cada agente puede conocer el conjunto de acciones disponibles de el otro agente , ambos agentes pueden saber por comunicación sus percepciones actuales o pueden inferir

las intenciones del otro basado en algún tipo de conocimiento compartido. En general en un sistema multi-agente cada miembro puede considerar el conocimiento de cada uno de los otros agentes en su propio proceso de decisión.

Las interacciones entre los agentes es comúnmente asociado con alguna forma de comunicación. Típicamente se considera el proceso de comunicación en un sistema multi-agente como un proceso de dos vías, donde todos los agentes pueden ser potencialmente receptores o emisores de mensajes . La comunicación puede ser utilizado en diferentes formas por ejemplo para coordinación entre agentes cooperativos(control de formación de robots o vehículos) o para una negociación entre agentes . Esto además plantea la cuestión de que protocolos de comunicación que se debe usar para el intercambio de información con el fin de lograr consensos en tiempo y forma [89].

Alguno de los beneficios de usar sistemas de multi-agentes en sistemas extensos son [90]:

- Mayor velocidad y eficiencia debido al computo paralelo
- Robustez y confiabilidad en el sentido que el sistema completo puede sufrir una degradación menor cuando uno de sus agentes falla.
- Flexibilidad y escalabilidad ya que es fácil agregar nuevos agentes al sistema.
- Desarrollo y rehuso, es mas fácil el desarrollar y mantener un sistema modular que un sistema monolítico.

2.2. Aprendizaje por reforzamiento

Los sistemas multi-agentes que emplean aprendizaje por reforzamiento (MARL) han sido utilizados en una variedad de problemas, la mayoría en simulación pero también en algunas tareas especificas de la vida real. Los problemas simulados dominan ya que los resultados obtenidos en entornos simples y controlados son mas fáciles de analizar, buscando obtener una visión mas profunda del problema, además que en aplicaciones experimentales la escalabilidad y robustez a observaciones imperfectas son necesarias y pocos algoritmos en

el campo de MARL exhiben estas propiedades. En aplicaciones de la vida real derivaciones directas de sistemas de un solo agentes son preferidas [91].

En el campo de control distribuido un conjunto de controladores autónomos e interactuantes actúan en paralelo en algún proceso. El control distribuido es una meta-aplicación para sistemas multi-agentes cooperativos: cualquier MAS es un sistema de control distribuido donde los agentes son los controladores, y su medio es el proceso a controlar. Por ejemplo en equipos de robots cooperativos ,los algoritmos de control están idenficados como los controladores y el medio junto con los sensores y actuadores son identificados como el proceso.

Algunos dominios particulares donde el control distribuido es aplicado son : en [92] presenta un esquema de control basado en aprendizaje por reforzamiento para una planta industrial de energía de carbón consistente en 4 agentes. En [93] usa aprendizaje por reforzamiento para un sistema multi-agente para obtener una eficiente política de control de señales de trafico. En [94] es utilizado MARL para el control de una red eléctrica .

Los equipos de robots son la aplicación mas popular en MARL encontrando una amplia rango de variaciones. Los robots usan MARL para adquirir un amplio espectro de habilidades, desde conductas básicas como navegación hasta conductas mas complejas como jugar fútbol soccer. En navegación cada robot tiene que encontrar su propia trayectoria desde una posición inicial hasta una posición objetivo ya sea fija o móvil , evitando obstáculos e interferencia con otros robots . En [95] se obtiene una solución óptima de aprendizaje para un equipo de robots que juegan soccer, manteniendo el aprendizaje de cada agente descentralizado en la medida de lo posible.

El barrido de área involucra navegación a través de un entorno bajo diversos propósitos : recuperación de objetos, cobertura por parte de los agentes de la mayor superficie del medio posible, y exploración donde los robots tienen que cubrir con su sensor la mayor cantidad de superficie del medio posible. Una variante de la anterior se encuentra en [96] donde un grupo de robots usa SLAM y MARL para decidir cuando fusionar las visiones de los sensores en un momento especifico y no hacerlo de manera inmediata.

En una variante de proceso depredador-presa,el grupo de robots intentan capturar objetivos móviles . En la transportación de objetos requiere acarrear un conjunto de objetos a

una posición final . La masa de los objetos pueden exceder las capacidades de transportación de un robot , por lo tanto requieren de coordinación para llevar acabo el objetivo final , en [97] usan estrategias combinadas de MARL y algoritmos genéticos para aprender una estrategia de cooperación en la transportación de objetos. Robots que juegan fútbol soccer son muy populares , es un banco de prueba para MARL , ya que para desarrollar esta actividad requiere la mayoría de las habilidades presentadas con anterioridad .

Los agentes en forma de software de comercio intercambian bienes en un supermercado electrónico a favor de una compañia o persona, usando mecanismos como negociación y subastas. Por ejemplo las competencias de agentes de comercio es un concurso simulado donde los agentes necesitan organizar paquetes de viaje por licitación de bienes tales como boletos de avión o reservaciones de hotel [98]. También se observan aplicaciones de MARL en comercio electrónico con el fin de mejorar la eficiencia en una negociación .

En algunos casos , el grupo de agentes cooperativos representa los intereses de una sola compañia o persona , y realizan diversas tareas en el proceso comercial, tal como comprar o vender, en otros casos los agentes interactúan en paralelo con los mercados .

Los agentes forman un equipo cooperativo donde ellos pueden desempeñar el papel de:

- Administradores de recursos: Cada agente administra un recurso y los agentes aprenden como responder de la mejor manera a requisiciones con el objetivo de optimizar una medida de desempeño.
- Cliente: Los agentes aprenden como seleccionar de la mejor manera recursos tal que una medida de desempeño es optimizada.

Como se menciono anteriormente un sistema multi-agentes puede ser usado para analizar problemas en una amplia variedad de dominios, incluyendo equipos de robots, control distribuido, manejo y administración de recursos, decisión colaborativa, minería de datos, telecomunicaciones, economía etc [51],[42],[53]. La complejidad de muchas tareas que se derivan de los dominios anteriores hacen dificil resolverlas con conductas pre-programadas en los agentes. De esta manera los agentes deben de encontrar una solución por si solos mediante aprendizaje.

Aunque los agentes en un sistema multi-agentes pueden ser dotados con un comportamiento diseñado con antelación, los agentes a menudo necesitan aprender nuevas conductas de tal manera que el rendimiento de los agentes gradualmente sea perfeccionado. Lo anterior usualmente es requerido ya que la complejidad del entorno y de las tareas asignadas a los agentes hacen que un diseño de las conductas a priori sea una asignación dificil o en ocasiones imposible de cumplir.

Las técnicas de aprendizaje por reforzamiento se han vuelto populares en los últimos años, gracias a que con ellos se pueden manejar problemas donde un comportamiento preprogramado de los agentes no es el mas adecuado, dando como resultado que los agentes aprendan por prueba y error mediante la interacción con el medio y con los demás agentes. El objetivo del aprendizaje por reforzamiento para sistemas multi-agentes es maximizar una función de recompensa [4].

En cada paso de aprendizaje , los agentes perciben la respuesta del entorno a las acciones tomadas por estos, lo que motiva al medio ambiente a transitar a un nuevo estado , la calidad de esta transición es evaluada mediante la función de recompensa , por lo tanto, a los agentes no se les dice que acciones deben de ser tomadas si no solo cuales acciones tienen la mayor recompensa. Es de tomar en cuenta que esta técnica es menos informativa que un método supervisado como las redes neuronales.

El uso del algoritmo llamado Q-learning propuesto originalmente Watkins [5] , puede simplificar la implementación del aprendizaje por reforzamiento, por medio de esta técnica se asume que el entorno puede ser descrito por un conjunto de estados y que los agentes desarrollan una cantidad limitada de acciones en cada paso de aprendizaje,además, se toma como cierto que el entorno es estático, al final del algoritmo se logra la convergencia hacia las acciones óptimas de cada agente con la idea de lograr el objetivo final. En aplicaciones reales , los agentes no cuentan con información completa acerca del entornos o de los cambios generados en el mismo , por lo tanto las técnicas tradicionales de aprendizaje por reforzamiento deben de ser complementadas para poder lidiar con esta problemática.

2.3. Aprendizaje por reforzamiento para un solo agente

El modelo para el aprendizaje por reforzamiento para un solo agente es llamado proceso de decisión de Markov:

Definición 2.1 *Un proceso finito de Markov para un agente es una tupla* (S, A, f, ρ)*:*

$$\begin{aligned} f &: S \times A \times S \to [0,1] \\ \rho &: S \times A \times S \to \mathbb{R} \end{aligned} \tag{2.1}$$

donde S *es el conjunto finito de estados en el entorno,* A *es el conjunto finito de acciones disponibles para el agente,* f *es la función de probabilidad de transición de estados y* ρ *es la función de recompensa la cual se asume acotada [6].*

El estado actual del medio el cual el agente observa es definido como $s_t \in S$, el cual describe al estado en cada paso de tiempo discreto t , el agente observa el estado y toma una acción definida como a_t. Como resultado el entorno cambia su estado a algún $s_{t+1} \in S$ de acuerdo a la función de probabilidad de transición de estados f , la probabilidad de acabar en el estado s_{t+1} después que la acción a_t es ejecutada en el estado s_t es $f(s_t, a_t, s_{t+1})$.

El agente recibe una recompensa escalar $r_{t+1} \in \mathbb{R}$, de acuerdo a la función de recompensa $\rho : r_{t+1} = \rho(s_t, a_t, s_{t+1})$. Esta recompensa evalúa el efecto inmediato de la acción a_t , es decir, la transición desde el estado s_t al estado s_{t+1} .Esta recompensa refleja que tan productiva fue la acción tomada a_t. Sin embargo esta señal no dice nada acerca de los efectos a largo plazo de esta acción tomada.

Para sistemas deterministicos, la función de probabilidad de transición de estados f y la función de recompensa ρ son reemplazadas por una función de transición mas simple

$$\begin{aligned} \bar{f} &= S \times A \to S \\ \bar{\rho} &= S \times A \to \mathbb{R} \end{aligned}$$

Donde la recompensa es completamente determinada por el estado actual y la acción actual $r_{t+1} = \rho(s_t, a_t)$. Algunos procesos de decisión de Markov tienen estados terminales

los cuales son estados donde una vez alcanzados no pueden ser abandonados en el futuro, se toma que todas las recompensas recibidas en un estado terminal se toman como cero [18]. En tal caso, el proceso de aprendizaje es usualmente separado en distintos episodios, los cuales son trayectorias comenzando desde algún estado inicial y finalizando en un estado terminal.

El comportamiento del agente es descrito por su política π, la cual especifica como el agente escoge sus acciones en un estado determinado del medio. La política π puede ser deterministica :

$$\bar{\pi} = S \rightarrow A$$

o estocastica:

$$\pi = S \times A \rightarrow [0, 1]$$

Una política es llamada estacionaria si esta no cambia en el tiempo. El objetivo final del aprendizaje por reforzamiento es encontrar una política π, para todo estado s que maximice el retorno R :

$$R^{\pi} = E\left\{ \sum_{t=0}^{\infty} \gamma^k r_{t+1} \middle| s_0 = s, \pi \right\} \tag{2.2}$$

donde $\gamma \in [0, 1)$ es el factor de descuento, la expresión anterior es tomada sobre la probabilidad de transición de estados sobre la política π, debemos notar que R representa la recompensa acumulada por el agente en el largo plazo. Existen otras formas de definir el retorno R en función de la actividad realizada [19]. El factor de descuento γ puede ser considerado como codificación de la incertidumbre creciente acerca de la recompensa que sera obtenida en el futuro o como un medio para acotar la suma en (2.2) de otro modo crecería de manera no acotada [8].

Por lo tanto el objetivo del agente es maximizar su rendimiento a largo plazo caracterizado por el retorno R solo recibiendo la realimentación acerca de su inmediata actuación en forma de la señal de recompensa r. Una forma de obtener el anterior resultado es por medio del calculo de una función óptima estado-acción de valor llamada funcion Q (Q-function)

$$Q^h : S \times A \rightarrow \mathbb{R} \tag{2.3}$$

la cual da un retorno R esperado dada una política π comenzando desde cualquier par estado-acción:

$$Q^{\pi}(s,a) = E\left\{ \sum_{t=0}^{\infty} \gamma^{k} r_{\,t+1} \middle| \, s_0 = s, a_0 = a, \pi \right\} \tag{2.4}$$

La función Q óptima es definida como Q^*

$$Q^{*}\left(s,a\right) = \max_{\pi} Q^{\pi}\left(s,a\right) \tag{2.5}$$

La cual satisface la ecuación de optimalidad de Bellman:

$$Q^{*}\left(s,a\right) = \sum_{\acute{s}\in S} f\left(s,a,s^{'}\right)\left[\rho\left(s,u,s^{'}\right) + \gamma \max_{a'} Q^{*}\left(s^{'},a^{'}\right)\right] \quad \text{para toda } s \in S \;\; a \in A \tag{2.6}$$

La expresión (2.6) señala que el valor óptimo al tomar la acción a en el estado s es la recompensa inmediata esperada mas el esperado valor óptimo esperado desde el siguiente estado. Una vez que Q^* esta disponible, una política de acciones óptima puede ser calculada por medio de escoger en cada estado una acción con el mas grande valor Q óptimo

$$\bar{\pi}^{*}\left(x\right) = \arg\max_{a} Q^{*}\left(s,a\right) \tag{2.7}$$

Cuando múltiples acciones alcanzan el mas grande valor de Q , cualquiera de ellos puede ser escogido y la política se mantiene óptima. Una política que maximice una función Q en esta forma se dice que es codiciosa (greedy) ,de este modo, el proceso para obtener una política óptima comienza encontrando Q^* , luego calcular una política codiciosa en Q^* ,es decir, una política que maximice la función Q óptima , a partir de este momento se dice que el agente tiene el conocimiento de la mejor combinación de acciones para realizar la tarea encomendada.

Existe una gran cantidad de algoritmos disponibles para el aprendizaje por reforzamiento para el caso de un solo agente, métodos free-model basados en la estimación en línea de funciones de valor [9],[10],[11], métodos basados en programación dinámica [12] y métodos basados en aprendizaje de modelos el cual estima modelos del sistema (puede ser considerado como modelos la función de transición de estados f, la función de recompensa ρ) [13],[14].

Uno de los métodos mas populares en el aprendizaje por reforzamiento para un solo agente es Q-Learning [5], en donde transforma la expresión (2.6) en un procedimiento de aproximación iterativa. Q-Learning comienza con una arbitraria función Q, observa las transiciones $(s_t, a_t, s_{t+1}, r_{t+1})$ y después de cada transición actualiza la función Q con :

$$Q_{t+1}(s_t, a_t) = Q_{t+1}(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q_t \left(s_{t+1}, a' \right) - Q_t \left(s_t, a_t \right) \right] \tag{2.8}$$

El termino entre corchetes es llamado una diferencia temporal [6], es decir la diferencia entre la estimación actual $Q_t \left(s_t, a_t \right)$ del valor óptimo de la función Q de (s_t, a_t) y de la estimación actualizada $r_{t+1} + \gamma \max_{a'} Q_t \left(s_{t+1}, a' \right)$. Este nuevo estimado es una muestra del lado derecho de la ecuación de Bellman (2.6), aplicado a Q_t en el par estado-acción (s_t, a_t).

En esta muestra, s' es reemplazada por el siguiente estado observado s_{t+1}, y $\rho \left(s_t, a_t, s' \right)$ por la recompensa observada r_{t+1}. El parámetro de aprendizaje $\alpha \in (0, 1]$ puede ser variante en el tiempo y usualmente decrece con el tiempo [15].

La secuencia Q_t converge a Q^* bajo las siguientes condiciones [5],[16],[17]:

- Distintos valores de la función Q son guardados y actualizado para cada par estado-acción.
- La sumatoria $\sum_{t=0}^{\infty} \alpha$ es finita.
- Asintóticamente todas los pares estado-acción son visitadas de manera infinita.

El tercer punto puede ser satisfecho si el agente se mantiene intentando todas las acciones en todos los estados que tengan probabilidad no cero de suceder. A este requerimiento se le denomina exploración, el cual puede ser hecho de varias formas , una de ellas es escogiendo en cada paso una acción aleatoria con probabilidad $\varepsilon \in (0, 1)$ y escogiendo una acción codiciosa (greedy) con probabilidad $\varepsilon - 1$, de esta manera obtenemos una exploración codiciosa. La probabilidad ε generalmente decrece con el tiempo.

Otra opción es usar el procedimiento de exploración de Boltzman, el cual en estado s se selecciona la acción a con una probabilidad:

$$h(, s, a) = \frac{e^{Q(s,a)/\tau}}{\sum_{\bar{a}} e^{Q(s,a)/\tau}} \tag{2.9}$$

donde $\tau > 0$ controla el grado de aleatoriedad de la exploración. Cuando $\tau \to 0$ la expresión (2.9) se convierte en un equivalente a la expresión (2.7). Cuando $\tau \to \infty$ la selección de la acción se vuelve aleatoria pura. Para $\tau \in (0, \infty)$, la acción mejor valuada tiene una mayor oportunidad de ser escogida que las acciones que tienen valores menores.

2.4. Aprendizaje por reforzamiento para sistemas multi-agentes

En juegos estocasticos totalmente cooperativos el retorno común R puede ser conjuntamente maximizado por todos los agentes, sin embargo, en otros casos el retorno R de los agentes son en general diferentes y están correlacionados no pudiendo ser maximizados de forma independiente.

La mayoría de los algoritmos propuestos en sistemas multi-agentes incorporan la estabilidad en las dinámicas de aprendizaje y la adaptación a los cambios en las conductas de los otros agentes en sus propuestas. De esta manera la estabilidad del algoritmo asegura la convergencia a una política $\boldsymbol{\pi}$ mientas que la adaptación asegura que el desempeño se mantiene o se mejora conforme los otros agentes cambian su política de acciones.

Los objetivos finales planteados a los agentes generalmente formulan condiciones para juegos estáticos en términos de estrategias y recompensas. Algunos de los objetivos pueden ser extendidos a juegos dinámicos, mediante el requerimiento de que las condiciones impuestas sean satisfechas en todos los estados del juego dinámico, en este caso los objetivos de aprendizaje son formulados en términos de estrategias por etapas y retornos esperados en lugar de estrategias y recompensas[37].

La convergencia a un punto de equilibrio de un algoritmo es un requerimiento básico para estabilidad, lo que implica que las estrategias de los agentes deberían eventualmente converger a un punto de equilibrio coordinado, donde el punto de equilibrio Nash es el mas

frecuentemente usado en los algoritmos de aprendizaje [39]. La convergencia de los algoritmos es requerida para estabilidad y la propiedad de racionalidad es añadida como un criterio de adaptación hacia las estrategias de los otros agentes.

Para que un algoritmo sea considerado como convergente se requiere que el agente que esta aprendiendo concurra a una política de acciones estacionarias siempre y cuando los demás agentes estén usando un algoritmo de aprendizaje previamente definido [40]. La racionalidad en un algoritmo es definida como el requerimiento de que un agente converja a la mejor respuesta cuando los demás agentes se encuentran estacionarios en sus estrategias. Si todos los agentes en el sistema son racionales y convergentes estos concurrirán (en ocasiones de manera implícita) a un punto de equilibrio Nash [38].

Un punto de equilibrio Nash es una estrategia conjunta de tal manera que cada estrategia individual genera la mejor respuesta posible a las estrategias de los demás agentes. El punto de equilibrio Nash describe un estado en el cual ningún agente se puede beneficiar del cambio en su estrategia individual siempre que todos los demás agentes se mantengan en sus estrategias constantes. Cualquier juego estático tiene al menos un punto (pudiendo no ser único) de equilibrio Nash [36].

Otras propiedades de los algoritmos para aprendizaje por reforzamiento en sistemas multi-agentes están relacionados con la estabilidad y adaptación. Por ejemplo en aprendizaje con algoritmos del tipo oponente-independiente es relacionado con la estabilidad, la cual prioriza la convergencia a una estrategia estacionaria, sin importa lo que los demás agentes están haciendo. En los algoritmos del tipo oponente-conciente los agentes aprenden modelos de los otros agentes y reaccionan a las acciones emprendidas por estos en forma de una mejor respuesta posible [41].

La característica de predicción en los algoritmos de aprendizaje esta relacionado con la estabilidad, el cual es la capacidad del agente de aprender modelos precisos del comportamiento de los demás agentes. La estabilidad en los procesos de aprendizaje es un requisito necesario ya que la conducta de agentes estables es mas fácil de tratar y analizar dando garantías significativas de rendimiento. La adaptación a otros agentes en un algoritmo es también es un requisito deseable ya que las conductas de los demás agentes son generalmente impredecible.

Por lo anterior un algoritmo en MARL deberá garantizar cotas en medidas de adaptación y estabilidad además de los requerimientos asintóticos tradicionales [43].

La generalización del proceso de decisión de Markov en el aprendizaje por reforzamiento en sistemas multi-agente (MARL) es el llamado juego estocastico [7].

Definición 2.2 *El juego estocastico es una tupla* $(S, A_1, A_2, ..., A_n, f, \rho_1, \rho_2, ..., \rho_n)$ *donde :*

$$\begin{aligned} &\rho_i : S \times \mathbf{A} \times S \to \mathbb{R} \\ &f : S \times \mathbf{A} \times S \to [0, 1] \\ &\mathbf{A} = A_1 \times A_2 .... \times A_n \end{aligned} \tag{2.10}$$

donde n *es el numero de agentes,* S *es el conjunto finito de estados en el entorno,* A_i $i = 1, ..., n$ *son los conjuntos finitos compuestos por las acciones disponibles de cada agente, produciendo el conjunto de acciones conjuntas* $A = A_1 \times A_2 \times ... \times A_n$*,* ρ_i *es la función de recompensa para el agente* i *la cual se considera acotada y la función de probabilidad de transición de estados siendo* f*.*

En el caso de sistemas multi-agentes las transiciones de estado son el resultado de las acciones conjuntas $\mathbf{a}_t$ tomadas por todos los agentes en el tiempo de paso discreto t :

$$\mathbf{a}_t = \left[a_{1,t}^T, ..., a_{n,t}^T\right]^T, \mathbf{a}_t \in \mathbf{A}, a_{i,t} \in A_i$$

Las políticas

$$\pi_i = S \times A_i \to [0, 1]$$

forman juntas la política de acción conjunta $\boldsymbol{\pi}$. Ya que las señales de recompensa $r_{i,t+1}$ de los agentes depende de una acción conjunta realizada por todos los agentes, su retorno R para un sistema multi-agente depende lo la política conjunta:

$$R_i^{\boldsymbol{\pi}}(x) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_{\,i,t+1} \middle| s_0 = s, \boldsymbol{\pi} \right\} \tag{2.11}$$

La función Q de cada agente depende de la acción conjunta y de la política de acciones conjunta

$$
\begin{aligned}
&Q_i^{\boldsymbol{\pi}} : S \times \mathbf{A} \rightarrow \mathbb{R} \\
&Q_i^{\boldsymbol{\pi}}(s, \mathbf{a}) = E\left\{ \sum_{k=0}^{\infty} \gamma^k r_{\,i,k+1} \,\middle|\, s_0 = s, \mathbf{a}_0 = \mathbf{a}, \boldsymbol{\pi} \right\}
\end{aligned} \tag{2.12}
$$

En juegos estocasticos enteramente cooperativos las funciones de recompensa son las mismas para todos los agentes $\rho_1 = \rho_2 = ... = \rho_n$ lo que implica que los retornos R también son los mismos $R_1^{\pi} = R_2^{\pi} = ... = R_n^{\pi}$, por lo tanto todos los agentes tienen el mismo objetivo el cual es maximizar el retorno común.

En el caso de dos agentes $n = 2$ y $\rho_1 = -\rho_2$, los agentes tienen un objetivo opuesto y el juego estocastico es enteramente competitivo, este tipo de tarea también puede involucrar mas de dos agentes , sin embargo, en la literatura de aprendizaje por reforzamiento para juegos enteramente competitivos típicamente maneja el caso de solo dos agentes. Los juegos estocasticos que no son enteramente cooperativos o competitivos son llamados juegos mixtos.

El aprendizaje por reforzamiento en los sistemas multi-agentes puede ser considerado como una fusión de diferencias-temporales RL (temporal-differences RL), especialmente se puede considerar en este apartado el algoritmo Q-learning [5] . El tipo de tarea considerado por los algoritmos de aprendizajes puede ser clasificados de acuerdo al objetivo final de los agentes, pudiendo ser completamente cooperativo, completamente competitivo y tareas mixtas, es de notar que la mayoría de los algoritmos son diseñados para tareas estáticas es decir sin estados de transición o que trabajan por etapas.

Un juego estático es un juego estocastico sin señal de estado y sin dinámica donde

$$S = \varnothing$$

Un juego estático es definido por una tupla $(A_1, A_2, ..., A_n, \rho_1, \rho_2, ..., \rho_n)$ donde la función de recompensa solo depende de las acciones conjuntas

$$\rho_i : \mathbf{A} \rightarrow \mathbb{R}$$

En los juegos donde hay solo dos agentes se le denomina juegos de bimatrices ya que la función de recompensa de cada uno de los agentes puede ser representada como una matriz

de $|A_1| \times |A_2|$ donde las filas corresponde a las acciones del agente 1 y las columnas a las acciones del agente 2. Los juegos estáticos que son completamente competitivos son llamados de suma cero, debido a que la suma de las matrices de recompensas de cada uno de los agentes es una matriz cero. Un juego mixto estático son llamados juegos de suma general, ya que no existe una restricción en la suma de las recompensas de los agentes.

Un juego por etapas es un juego estático que emerge en un cierto estado del juego estocastico. La función de recompensa de la etapa en el estado s son las funciones Q del juego estocastico proyectado en el espacio de estados conjuntas donde el estado s es fijo. En general los agentes visitan el mismo estado de un juego estocastico múltiple veces, razón por la que se le llama juego repetido. En los juegos repetidos, los agentes pueden obtener información acerca del comportamiento de los otros agentes o acerca de las funciones de recompensa y de este modo realizar decisiones mas informadas.

En un juego estático o repetido, la política de acciones pierde el argumento de estado y se transforma en una estrategia :

$$\sigma_i = A_i \rightarrow [0, 1]$$

La estrategia de juego de un agente para un juego por etapas derivado de algún estado s de un juego estocastico es la proyección de su política λ_i en su espacio de acción A_i cuando el estado es fijo en s [35].

Si se define la mejor respuesta del agente i a un vector de estrategias de sus oponentes como σ_i^* que alcance la máxima recompensa esperada dado un vector de estrategias

$$E\{r_i | \sigma_1, ..., \sigma_i, ..., \sigma_n\} \leq E\{r_i | \sigma_1, ..., \sigma_i^*, ..., \sigma_n\} \text{ para toda } \sigma_i$$

por lo anterior el punto de equilibrio Nash es una estrategia conjunta $[\sigma_1^*, \sigma_2^*, ..., \sigma_n^*]$ de tal manera que cada estrategia individual σ_i^* es la mejor respuesta a las otras estrategias de los demás oponentes [36].

2.4.1. Cooperativas para sistemas multi-agentes.

En una tarea completamente cooperativa , todos los agentes tiene la misma función de recompensa ($\rho_1 = \rho_2 = ... = \rho_n$) y el objetivo de aprendizaje es maximizar el retorno

R común. Si un control centralizado estuviese disponible , la tarea se vería reducida a un proceso de decisión de Markov como en el caso de un solo agente. En este caso el objetivo de aprendizaje podría ser logrado obteniendo valores óptimos de acciones conjuntas mediante Q-learning:

$$Q_{t+1}(s_t, \mathbf{a}_t) = Q_{t+1}(s_t, \mathbf{a}_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q_t \left(s_{t+1}, \mathbf{a}' \right) - Q_t \left(s_t, \mathbf{a}_t \right) \right] \tag{2.13}$$

y luego usando una política codiciosa. Sin embargo los agentes son entidades que toman decisiones de manera independiente y el problema de coordinación emerge incluso si todos los agentes aprenden en paralelo la función Q óptima común usando la expresión (2.13). Podría parecer lo mas lógico que los agentes usaran políticas codiciosas aplicadas a la función óptima Q^* con el objetivo de maximizar el retorno común:

$$\pi_i^*(s) = \arg \max_{aj} \max_{a\,1,\ldots a_{i-1}, a_{i+1}, \ldots, a_n} Q^* \left(s, \mathbf{a} \right) \tag{2.14}$$

Sin embargo en ciertos estados, múltiples acciones conjuntas pueden ser óptimas. En la ausencia de mecanismos adicionales de coordinación, diferentes agentes podrían escoger distintas acciones óptimas conjuntas y como resultado la acción conjunta podría resultar sub óptima.

Existen diferentes tipos de algoritmos que evaden el problema de coordinación bajo diferentes supuestos, por ejemplo, el algoritmo Team Q-Learning [21]supera este problema asumiendo que las acciones conjuntas óptimas son únicas (el cual no es siempre cierto), por lo tanto, si todos los agentes aprenden la función Q común en paralelo con la expresión (2.13), los agentes pueden usar (2.14) para seleccionar las acciones conjuntas óptimas y maximizar el retorno R.

El algoritmo Distributed Q-learning [99], resuelve el problema de la cooperatividad sin asumir coordinación mediante una complejidad computacional similar a el caso de un solo agente, sin embargo, el algoritmo trabaja en problemas deterministicos con funciones de recompensas no negativas. Cada agente i mantiene una política local $\bar{\lambda}_i(s)$ y una función Q local $Q_i(s, a_i)$ la cual depende solo de la acción individual del agente. Los valores locales de la función Q son actualizadas solo cuando la actualización lleva a un incremento en el valor

de Q

$$Q_{i,t+1}(s_t, a_{i,t}) = \text{máx}\left\{Q_{i,t}(s_t, a_{i,t}), r_{t+1} + \gamma \,\text{máx}\, Q_{i,t}(s_{t+1}, a_i^{'}\right\} \tag{2.15}$$

Lo anterior asegura que el valor local de la función Q siempre capturara el máximo valor Q de la acción conjunta :

$$Q_{i,t}(s_t, a_i) = \underset{a_1,..,a_{i-1},a_{i+1},..,a_n}{\text{máx}} Q_t\left(s, \mathbf{a}\right) \text{ para toda } t$$

donde:

$$\mathbf{a} = \left[a_1, a_2, ..., a_n\right]^T \text{ con } a_i \text{ fija}$$

La política local es actualizada solo si la actualización nos lleva a una mejora en los valores Q.

$$\bar{\pi}_{i,t+1}\left(s_t\right) = \begin{cases} a_{i,t} \text{ si } \text{máx}_{\bar{a}_i} Q_{i,t+1}\left(s_t, \bar{a}_i\right) > \text{máx}_{\bar{a}_i} Q_{i,t}\left(s_t, \bar{a}_i\right) \\ \bar{\pi}_{i,t}\left(s_t\right) \quad \text{de otro modo} \end{cases} \tag{2.16}$$

Esto asegura que la política conjunta $[\bar{\pi}_{1,t}, ..., \bar{\pi}_{n,t}]$ sera siempre óptima con respecto al valor global Q_t.La mayoría de los algoritmos presentes en el aprendizaje por reforzamiento se basan en mediciones exactas del estado, lo anterior es recurrente en métodos de coordinación libre donde si en algún punto la percepción de los agentes difiere, nos puede llevar a que los agentes actualicen las funciones Q de manera distinta de tal manera que la consistencia de las políticas encontradas no pueden ser garantizadas.

Un método popular usado para superar el problema de la coordinación es el uso de los gráficos de coordinación [22], estos ayudan a simplificar la coordinación cuando la función global Q puede ser descompuesta en funciones Q locales que solo depende de las acciones de subconjuntos de agentes. Por ejemplo en un juego estocastico con 4 agentes la descomposición en funciones Q locales podría ser:

$$Q\left(s, \mathbf{a}\right) = Q_1\left(s, a_1, a_2\right) + Q_2\left(s, a_1, a_3\right) + Q_3\left(s, a_{3,}a_4\right)$$

La descomposición puede ser diferente para distintos estados, donde las funciones Q locales tienen la misma dimensión que las funciones Q globales.

La maximización de los valores Q generales es hecha por medio de una resolución de maximizaciones locales mas simples en términos de la función local Q y agregando sus soluciones. Bajo ciertas condiciones, la selección coordinada de una acción conjunta óptima es garantizada [23]

Los métodos de coordinación indirecta se inclinan hacia la selección de acciones que son mas probables de obtener una buena cantidad de retorno R. Esto conduce a los agentes hacia una selección de acciones coordinadas. La posibilidad de obtener buenos retornos R es evaluado por distintos métodos, por ejemplo usando modelos estimados de los otros agentes u observación de estadísticas de retornos R ocurrido en el pasado.

En el caso de tareas estáticas (donde el juego estocastico no tiene señal por parte del estado) existen distintos algoritmos disponibles para la coordinación indirecta, entre ellos tenemos el algoritmo Joint Action Learners (JAL) [24], la cual aprende los valores de las acciones conjuntas. El agente i aprende modelos de los otros agentes $j \neq i$ usando:

$$\bar{\sigma}_j^i(a_j) = \frac{C_j^i(a_j)}{\sum_{\bar{a}_j \in A_j} C_j^i(\bar{a}_j)} \tag{2.17}$$

donde $\bar{\sigma}_j^i$ es el modelo del agente i observado por el agente j , $C_j^i(a_j)$ es el conteo del numero de veces que el agente i observa al agente j tomando la acción a_j. El agente i tiene que observar las acciones tomadas por los otros agentes .

Otra opción es el algoritmo Frequency Maximum Q-value (FMQ) el cual esta basado en la frecuencia en la cual las acciones dieron buenos retornos en el pasado . El agente i usa un procedimiento de exploración similar a (2.9) y calcula los valores Q mediante:

$$\bar{Q}_i(a_i) = Q_i(a_i) + v\frac{C_{\text{máx}}^i(a_i)}{C^i(a_i)} r\,\text{máx}(a_i)$$

donde $r_{\text{máx}}(a_i)$ es la máxima recompensa observada después de tomar la acción a_i, mientas que $C_{\text{máx}}^i(a_i)$ cuenta cuantas veces esta recompensa ha sido observada, mientras que $C^i(a_i)$ cuenta cuantas veces la acción a_i ha sido tomada, siendo v un factor de ponderación. Mediante el incremento de los valores Q de las acciones que frecuentemente producen buenas recompensas en el pasado conduce al agente hacia la coordinación.

En una comparación con el caso de un solo agente donde se aplique el algoritmo de aprendizaje Q-learning, el único requerimiento adicional de computación en el algoritmo Frequency Maximum Q-value (FMQ) viene de mantener un conteo de las acciones . Sin embargo el algoritmo FMQ puede fallar en algunos problemas donde la función de recompensa es del tipo estocastico, el vector v debe de ser sintonizado en función de cada problema en particular, lo cual resulta complicado en general.

Para el caso de tareas dinámicas la mayoría de los algoritmos construyen juegos virtuales en cada etapa del juego estocastico, en estos juegos virtuales, una acción óptima conjunta es recompensada con el valor de 1 y el resto de las acciones con el valor de 0. El algoritmo es introducido de manera que lleva a los agentes hacia las mas recientes acciones óptimas seleccionadas, garantizando la convergencia hacia una acción óptima coordinada para el juego virtual y por lo tanto también a una acción conjunta coordinada para el juego original, un ejemplo es el algoritmo Optimal Adaptive Learning (OAL) [25], este algoritmo probablemente converge a una política conjunta óptima en casi cualquier juego estocastico enteramente cooperativo.

Lo anterior lleva a un incremento en la complejidad computacional donde cada agente debe estimar un modelo de un juego estocastico, modelos de los otros agentes y una óptima función de valor para el juego estocastico original.

Un enfoque generalizado utilizado para resolver el problema de la coordinación en un sistema multi-agentes es asegurarse que cualquier situación de decisión sea resuelto de la misma manera por todos los agentes usando algún tipo de negociación o explícita coordinación. Los mecanismos que usan esta técnica están basados en convenciones sociales, roles, comunicación entre otros, los cuales pueden ser utilizados en cualquier tipo de tarea [26].

Las convenciones sociales y los roles restringen la elección de acción de los agentes. Un rol restringe de antemano el conjunto de acciones disponible para un agente en el momento de seleccionar una acción a desarrollar[27]. Esto nos dice que alguno de los empates (donde acciones son igualmente óptimas) son prevenidos dado :

$$\pi_i^*(s) = \arg \max_{aj} \max_{a\,1,\ldots a_{i-1},a_{i+1},\ldots,a_n} Q^*(s, \mathbf{a}) \tag{2.18}$$

Las convenciones sociales codifican preferencias dadas de antemano hacia ciertas acciones conjuntas y ayuda a romper empates durante la selección de acciones. Si estas son adecuadamente diseñadas, los roles y las convenciones sociales eliminan completamente los empates en el proceso de selección de acciones. Una convención social simple se basa en un ordenamiento único de los agentes y sus acciones, donde estos dos ordenamientos deben de ser conocidos por todos los agentes. Combinando ambos nos lleva a un ordenamiento único de las acciones conjuntas y la coordinación esta asegurada si en la expresión (2.18) la primera acción conjunta en este ordenamiento es seleccionada por todos los agentes [28].

La comunicación puede ser usada para negociar elecciones de acciones entre agentes, ya sea usando comunicaciones de forma solitaria o en combinación con roles y convenciones sociales. Cuando son combinadas con las técnicas primeramente mencionadas, las consideraciones acerca de la comunicación pueden ser relajadas y simplificar su aplicación. Además de coordinar acciones, los agentes pueden comunicar una variedad de información incluyendo Q tablas de manera completa o parcial, mediciones de estado, recompensas, parámetros de aprendizaje entre otros [29].

También han sido investigado algunos enfoques donde la coordinación es aprendida por los agentes en lugar de ser programado a priori, estos agentes pueden aprender convenciones sociales [30], asignación de roles[31], estructuras de gráficos de coordinación junto con funciones locales Q [32].

2.4.2. Completamente competitivas

En un juego estocastico enteramente competitivo compuesto por dos agentes donde $\rho_1 = -\rho_2$, el principio min-max puede ser aplicado: maximizar el beneficio de un agente bajo el supuesto del peor escenario cuyo oponente siempre se esforzara por minimizarlo. El algoritmo minimax-Q [21],[33] emplea el principio de min-max para calcular las estrategias y valores, además de utilizar diferencias temporales similares al Q-learning con la intención de propagar

los valores a través de las transiciones de estados, para el agente 1 se tendría

$$\begin{aligned} \pi_{1,t}(s_t, \cdot) &= \arg m_1(Q_t, s_t) \\ Q_{t+1}\left(s_t, a_{1,t}, a_{2,t}\right) &= Q_t\left(s_t, a_{1,t}, a_{2,t}\right) + \alpha\left[s_{t+1} + \gamma \mathbf{m}_1(Q_t, s_{t+1}) - Q_t\left(s_t, a_{1,t}, a_{2,t}\right)\right] \end{aligned}$$

donde $\mathbf{m}_1$ es el retorno del agente 1

$$\mathbf{m}_1\left(Q, s\right) = \underset{h_1(s,\cdot)}{\text{máx}} \underset{a_2}{\text{mín}} \sum_{a_1} \pi_1\left(s, a_1\right) Q\left(s, a_1, a_2\right) \tag{2.19}$$

La estrategia estocastica del agente 1 en el estado s en el tiempo t se denota por $\pi_{1,t}(s_t, \cdot)$, con el punto en la posición del argumento correspondiente a la acción. El problema de optimización mostrado en la expresión (2.19) puede ser resuelto por medio de programación lineal [34]. El anterior método minimax-Q es un algoritmo en donde si la optimización min-max tiene múltiples soluciones o estrategias, cualquiera de ellas puede alcanzar al menos el retorno minimax, sin importa las acciones del oponente. Sin embargo, si el oponente es subóptimo (cuando no toma siempre la acción que perjudicaría en mayor proporciona al oponente) y el otro agente tiene un modelo de la política del oponente, este agente podría tener un mejor retorno que el expresado en (2.19).

Un modelo del oponente puede ser aprendido mediante una extensión de (2.17) para el caso de múltiples estados:

$$\pi_j^i\left(s, a_j\right) = \frac{C_j^i\left(s, a_j\right)}{\sum_{\bar{a}_j \in A_j} C_j^i\left(s, \bar{a}_j\right)} \tag{2.20}$$

donde $C_j^i\left(s, a_j\right)$ es el contador de las veces que el agente i observa al agente j tomando la acción a_j en el estado s [3].

En juegos estocasticos mixtos, ningún tipo de restricciones son impuestas en las funciones de recompensas ρ de cada uno de los agentes. Este modelo de tareas es mas apropiado por agentes que son motivados por sus propios intereses pero no necesariamente compiten con otros agentes. Los conceptos de equilibrio tienen gran influencia en los algoritmos dedicados al aprendizaje por reforzamiento para sistemas multi-agentes en las tareas mixtas. Cuando una cantidad múltiple de puntos de equilibrio exista en un juego estocastico emerge

el problema de elegir uno, surgiendo la necesidad en los agentes de seleccionar de manejara consistente su parte en el mismo punto de equilibrio.

Un numero significante de algoritmos han sido desarrollados para tareas estáticas, juegos repetidos y juegos de suma cero. En juegos repetidos el problema de aprendizaje no es estacionario debido a las dinámicas de las conductas de los agentes. Varias técnicas de aprendizaje para un solo agente son utilizados en sistemas multi-agente con tareas mixtas, por la razón que los métodos de un solo agente no hacen ningún tipo de conjeturas previas acerca de la tarea a desarrollar y de este modo son aplicables a juegos estocasticos [38].

Los algoritmos diseñados para un solo agente tales como Q-Learning puede ser directamente aplicado a el caso de sistemas multi-agentes, donde los agentes aprenden funciones Q que solo depende de la acción actual del agente mediante el uso de la expresión (2.8) y sin ser consciente de las acciones de los otros agentes. La no estacionalidad de los sistemas multi-agentes (debido a las conductas independientes de cada uno de los agentes) inválida las garantías teóricas que proporciona el algoritmo Q-Learning, pero a pesar de estas limitantes este enfoque ha encontrado muchas aplicaciones en sistemas multi-agentes debido a su simplicidad de implementación[52],[44].

Muchos algoritmos que son independientes a las acciones de los demás agentes comparten una estructura común con el algoritmo Q-learning, donde políticas y valores de estado son calculados mediante teoría de juegos aplicado a juegos por etapas :

$$\begin{aligned} \pi_{i,t}(s,\cdot) &= \text{solve}_i\{Q_{\cdot,t}(s_t,\cdot)\} \\ Q_{i,t+1}(s_t,\mathbf{u}_t) &= Q_{i,s}(s_t,\mathbf{u}_t) + \alpha\left[r_{i,t+1} + \gamma\,\text{eval}_i\{Q_{\cdot,t}(s_{t+1},\cdot)\} - Q_{i,s}(s_t,\mathbf{u}_t)\right] \end{aligned}$$

Donde $\{Q_{\cdot,t}(s_t,\cdot)\}$ denota el juego por etapas que emerge en el estado s dado por todas las funciones Q de los agentes en el tiempo t, el termino solve_i regresa la parte de algún tipo de estrategia del agente i y eval_i da el esperado retorno del agente i dada la anterior estrategia. El objetivo es la convergencia a un equilibrio en cada estado del sistema.

Capítulo 3

Planeación de multi-agentes vía aprendizaje por reforzamiento neuronal

La planeación de trayectoria se basa en generar una trayectoria desde un punto inicial hasta un punto final en función de ciertas restricciones impuestas por el medio o por las tareas encomendadas a los agentes. Dado un conocimiento parcial acerca del ambiente y de la posición del objetivo final, la planeación de trayectoria engloba la habilidad de los agentes de actuar basado en el conocimiento adquirido para alcanzar su posición objetiva final mediante la identificación de una trayectoria, además los agentes deben realizar las tareas encomendadas de la manera mas eficiente y segura posible. La planeación de trayectoria es un problema estratégico ya que los agentes deben de decidir que acciones tomar a largo plazo con el objetivo de completar sus tareas asignadas.

El problema de generación de trayectoria en el caso de un solo agente ha sido ampliamente abordado en la literatura, dando como resultado algoritmos ampliamente probados y con una robusta base teórica, sin embargo, en los sistemas multi-agentes aun es un problema abierto, donde la manera independiente en que las conductas de los agentes se desarrolla atenta contra los supuestos de estacionalidad en que los algoritmos de aprendizaje por reforzamiento para

un solo agente están basados.

Un sistema multi-agente incluye varias entidades inteligentes llamados agentes en un entorno determinado. Cada agente tiene su propia forma de conducirse persiguiendo su propio objetivo final, en el caso de las tareas cooperativas, cada agente genera su propio comportamiento con el fin de conseguir un objetivo común, surgiendo la necesidad de coordinarse con los demás agentes del medio [1].

Un importante beneficio de utilizar sistemas multi-agentes es posible modelar comportamientos de la vida común, de este modo los agentes aprenden nuevas conductas de tal manera que pueden adaptarse y predecir el funcionamiento de los sistemas mediante un control descentralizado [45].

Las tareas que comúnmente realizan los sistemas multi-agentes se encuentran comúnmente en diversas actividades de la vida diaria y de la industria tales como robots móviles , tareas de logística, diseño de juegos entre otros [2]. Por lo que el estudio de este tema en sistemas multi-agentes es de gran interés, no solo por la aplicaciones mencionadas anteriormente si no también por las posibles extensiones que podrían derivarse de este tema en particular.

Una gran cantidad de enfoques han sido propuestos para resolver el problema de la planeación de trayectoria, el método de grafos en red (grid graph) es aplicado para el diseño de una trayectoria que busca desempeñar una tarea determinada mediante la búsqueda heurística en el grafo[46], también los grafos indirectos son utilizados para sistemas multi-agentes donde el tiempo de ejecución y la calidad de la solución obtenida son acotadas [47], otra opción que muestra una convergencia mas rápida es el algoritmo de arboles aleatorios el cual es extendida de una manera adecuada para la generación de trayectorias [48], los algoritmos genéticos han mostrado gran utilidad en la generación de trayectorias [49], así como los métodos que usan lógica difusa [50].

El aprendizaje por reforzamiento es una de los métodos mas populares para el aprendizaje no supervisado en los sistemas multi-agentes (MARL), en donde el agente no se hace del conocimiento de que acciones debe de tomar si no que ellos solo saben que acciones les genera una mayor recompensa inmediata. La recompensa que se genera después de realizar

una determinada acción es menos informativa que un método supervisado en las situaciones similares. En virtud que en aplicaciones reales ningún agente tiene una información completa acerca de la situación del entorno, además de que en el medio en donde se desarrolle la acción puede resultar modificado, los algoritmos clásicos para MARL deben de ser modificados para poder lidiar con las dinámicas del entorno.

En este capitulo proponemos la planeación de trayectoria para sistemas multi-agentes usando una técnica basada en dos etapas,en la primera etapa usamos un algoritmo de aprendizaje por reforzamiento clásico (WoLF-PHC) el cual nos dará como resultado una política de acciones que generara una trayectoria disponible para los agentes bajo el supuesto de una configuración determinada del ambiente de trabajo [3],en esta etapa los agentes exploran el entorno desconocido y colectan información estado-acción, obteniendo como resultado una Q-tabla con las acciones óptimas en estados determinados, en la segunda etapa usaremos las características de aproximación no lineal de las redes neuronales para aproximar las acciones de los agentes cuando el medio sufre variaciones en su configuración o los agentes son expuestos a distintas condiciones iniciales del problema, usaremos la información obtenida en la primera etapa en forma de Q-tabla para entrenar la red neuronal. Finalmente cuando los agentes se encuentren en estados desconocidos, una política conjunta óptima que actuara como señal de control es dada en forma de una política codiciosa a partir de la generalización realizada por la red neuronal .

3.1. Aprendizaje por reforzamiento en entorno desconocido

Es sabido que los algoritmos que utilizan el aprendizaje por reforzamiento necesitan de el conocimiento de todos los pares estados-acciones presentes en el entorno para poder obtener una política de acciones óptimas, por lo tanto estas estrategias sufren del problema de la dimensionalidad, el cual es causado por el crecimiento exponencial del espacio discreto, representado por el numero de estados y variables de acción de los agentes, por lo que la

dimensión de un sistema esta en función de la cantidad de estados presentes en el entorno y de las acciones disponibles para cada agente.

Ya que los algoritmos tradicionales de aprendizaje por reforzamiento estiman los valores para cada posible estado discreto, lo anterior nos lleva a un aumento en la complejidad computacional necesaria para lograr una convergencia hacia una política de acciones óptimas. La complejidad computacional en el aprendizaje por reforzamiento en los sistemas multi-agentes se agudiza en función de el numero de agentes presentes, esto es debido a que cada agente añade sus propias variables al espacio conjunto de acciones-estado, lo que hace mas severo el problema de la dimensionalidad en el caso multi-agente que en el caso de un solo agente.

Por lo expuesto anteriormente, el método de kernel smoothing en una primera propuesta sera usado, además de una red neuronal en una segunda instancia, donde ambas metodologias serán utilizadas para estimar las acciones mejor valuadas en los estados del entorno que no fueron visitados por los agentes durante el proceso de aprendizaje por reforzamiento, aproximando acciones sub óptimas cuando las condiciones iniciales del problema han sido cambiadas o el entorno ha sido modificado. Lo anterior evitara volver a ejecutar el algoritmo de aprendizaje minimizando la complejidad computacional. La red neuronal sera entrenada con los datos obtenidos por el algoritmo de aprendizaje por reforzamiento ejecutado en una primera etapa.

El flujo de trabajo propuesto se muestra en la Figura 3.1, donde los sensores obtienen la información proveniente del entorno, esta información es alimentada al algoritmo de aprendizaje por reforzamiento y al kernel.

En la etapa de adquisición de datos el algoritmo de aprendizaje por reforzamiento para sistemas multi-agentes utilizado es una versión modificada del algoritmo WoLF-PHC (Win or Learn Fast- Policy Hill Climbing) [3], este algoritmo es considerado racional ya que requiere que la conducta de un agente converja a una mejor respuesta cuando los otros agentes se mantienen estacionarios en su conducta. El algoritmo WoLF-PHC posee buenas cualidades de convergencia y es capaz de jugar estrategias mixtas, la cual implica que los agentes seleccionen sus acciones de acuerdo a una distribución de probabilidad sobre las acciones

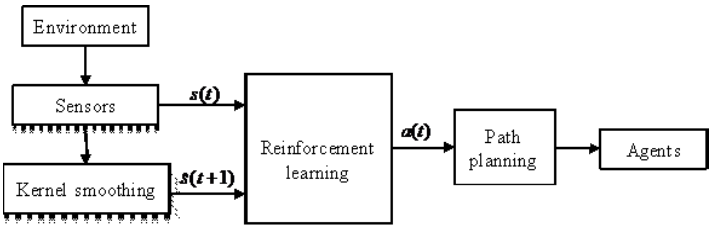

Figura 3.1: Flujo de trabajo del algoritmo propuesto

disponibles en lugar de estrategias puras donde los agentes seleccionan acciones de una manera determinista.

El algoritmo WoLF-PHC puede ser usado incluso en situaciones donde los agentes están utilizando otros algoritmos de aprendizaje es decir que son agentes heterogéneos, además no es necesario un conocimiento a priori acerca del modelo del entorno y la tarea a desarrollar por parte de los agentes, los cuales son expresados en forma de funciones de recompensa y funciones de transiciones de estados.

Este algoritmo puede ser considerado como una extensión del algoritmo Q-Learning desarrollado originalmente para un solo agente. En el algoritmo Q-learning la experiencia del agente consiste en una serie de episodios, donde en el n-esimo episodio el agente [5] :

- Observa su estado actual x_n
- Selecciona y ejecuta una acción a_n
- Observa el siguiente estado generado y_n
- Recibe una recompensa inmediata r_n
- Ajusta sus valores Q_{n-1} usando un factor de aprendizaje α_n de acuerdo a:

$$Q_n(x,a) = \begin{cases} (1-\alpha_n)\, Q_{n-1}(x,a) + \alpha_n \left[r_n + \gamma V_{n-1}(y_n)\right] \\ \text{si } x = x_n \text{ y } a = a_n \\ Q_{n-1}(x,a) \\ \text{de otra manera} \end{cases}$$

donde $\gamma \in [0,1)$ es el factor de descuento

$$V_{n-1}(y) = \max_b \left[Q_{n-1}(y,b)\right]$$

es la mejor respuesta que el agente puede realizar desde un estado y.

En los estados iniciales del proceso de aprendizaje, los valores Q obtenidos podrían no reflejar de una manera exacta la política final, por lo que es requisito que los valores iniciales $Q_0(x,a)$ para todos los estados y acciones sean dadas, por lo general se establecen en valores de cero. De esta manera esta descripción asume una representación en forma en forma de tabla de búsqueda para los valores $Q_n(x,a)$, para otro tipo de representaciones se ha demostrado que podría no haber una correcta convergencia hacia los valores óptimos [5].

El algoritmo PHC (Police Hill Climbing) utilizado en los sistemas multi-agentes, los valores Q obtenidos son preservados como en el caso de un solo agente cuando se usa Q-Learning bajo políticas mixtas. En el algoritmo PHC la política es mejorada mediante el incremento de la probabilidad de seleccionar la acción mejor valuada en un estado determinado de acuerdo a una razón de aprendizaje $\delta \in [0,1)$, es de notar que cuando $\delta = 1$ el algoritmo PHC es similar al algoritmo Q-Learning, ya que en cada paso de aprendizaje la política se mueve hacia una política codiciosa la cual ejecuta la acción mejor valuada con una probabilidad de 1, a pesar que el algoritmo PHC es racional y puede manejar políticas mixtas presenta problemas de convergencia cuando se enfrentan los agentes a tareas mixtas es decir tareas que no son completamente cooperativa ni completamente competitivas [3].

El algoritmo PHC (Police Hill Climbing) propuesto por M. Bowling es dado por :

1. Sea $\alpha \in (0,1]$, y el factor de aprendizaje $\delta \in (0,1]$, se inicializa :

$$Q(s,a) \longleftarrow 0, \pi(s,a) \longleftarrow \frac{1}{|A_i|} \tag{3.1}$$

donde $\pi(s,a)$ es la probabilidad de escoger la acción a en el estado s, $|A_i|$ es la cardinalidad del conjunto A.

2. Repetimos

   - Para el estado s, seleccionamos la acción a de acuerdo a una estrategia mixta $\pi(s)$ con una explotación adecuada . En cada paso de aprendizaje una acción aletearía con probabilidad $\varepsilon \in (0,1)$ es escogida . Luego una acción codiciosa con el mayor valor Q con una probabilidad $(1-\varepsilon)$ es calculada.
   - Se obtiene la recompensa r y se observa el siguiente estado $\acute{s}$

$$Q(s,a) \longleftarrow (1-\alpha)\, Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(\acute{s}, \acute{a}) \right) \tag{3.2}$$

   - Se mantiene π mas cercana a una política óptima

$$\pi\,(s,a) \longleftarrow \pi\,(s,a) + \Delta_{sa} \tag{3.3}$$

   - Manteniéndose restringida a una probabilidad de distribución

$$\Delta_{sa} \begin{cases} = -\delta_{sa} & \text{si } a \neq \arg\max_{\acute{a}} Q(s,\acute{a}) \\ = \sum_{\acute{a} \neq a} \delta_{sa'} & \text{de otro modo} \end{cases} \tag{3.4}$$

$$\delta_{sa} = \min \left(\pi(s,a), \frac{\delta}{|A_i| - 1} \right) \tag{3.5}$$

Con el objetivo alentar las propiedades de convergencia sin sacrificar las propiedades de racionalidad, el algoritmo PHC es modificado en una nueva versión llamado WoLF-PHC (Win or Learn Fast -Policy Hill Climbing) [3], donde utiliza un factor de aprendizaje variable δ el cual ayuda en la convergencia, otorgando mas tiempo a los agentes de adaptarse a los cambios en las estrategias de los otros agentes, que en una primera instancia podría ser benéfico y al mismo tiempo habilita al agente de adaptarse mas rápidamente a los cambios en las estrategias de los otros agentes cuando estas trabajan en detrimento del primero.

El algoritmo WoLF-PHC usa dos parámetros de aprendizaje $\delta_L > \delta_w$ los cuales son usados para actualizar las políticas de los agentes mediante la comparación de que si el valor esperado es mayor que el valor esperado en una política promedio, esta política promedio es usada como una aproximación de un equilibrio, de esta manera es posible lograr la convergencia hacia políticas óptimas en problemas con tareas mixtas. Se considera que los agentes convergen a una política estacionaria cuando los otros agentes no se mueven hacia otras políticas [55].

El algoritmo WoLF-PHC para el agente i es:

1. Sea $\alpha \in (0, 1]$, el factor de aprendizaje $\delta_l > \delta_w \in (0, 1]$, se inicia:

$$Q(s, a) \longleftarrow 0, \pi(s, a) \longleftarrow \frac{1}{|A_i|} \tag{3.6}$$

   donde $\pi(s, a)$ es la propiedad de escoger la acción a en el estado s, $|A_i|$ es la cardinalidad del conjunto A.

2. Repite

   - Para el estado s, se selecciona la acción a de acuerdo a una estrategia mixta $\pi(s)$ con una explotación adecuada. En cada paso de aprendizaje una acción aletearía con probabilidad $\varepsilon \in (0, 1)$ es usada. Una acción codiciosa con el mayor valor Q con una probabilidad $(1 - \varepsilon)$ es seleccionada.
   - Se obtiene la recompensa r y el siguiente estado $\acute{s}$

$$Q(s, a) \longleftarrow (1 - \alpha) Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(\acute{s}, \acute{a}) \right) \tag{3.7}$$

   - Actualiza el promedio de las políticas $\bar{\pi}$

$$\begin{aligned} C(s) &\longleftarrow C(s) + 1 \\ \bar{\pi}(s, \acute{a}) &\longleftarrow \tfrac{1}{C(s)} \left(\pi(s, \acute{a}) - \bar{\pi}(s, \acute{a}) \right) \end{aligned} \tag{3.8}$$

   donde $\forall \acute{a} \in A_i$,

- Se mueve la política π hacia una política óptima con respecto a la Q-table

$$\pi(s,a) \longleftarrow \pi(s,a) + \Delta_{sa} \tag{3.9}$$

donde las restricciones y la probabilidad de distribución son:

$$\begin{cases} \Delta_{sa} = -\delta_{sa} & \text{si } a \neq \arg\max_{\acute{a}} Q(s,\acute{a}) \\ \Delta_{sa} = \sum_{\acute{a}\neq a} \delta_{sa'} & \text{de otro modo} \end{cases} \tag{3.10}$$

con $\delta_{sa} = \min\left(\pi(s,a), \frac{\delta}{|A_i|-1}\right)$

$$\delta \begin{cases} = \delta_w & \text{si } \sum_{a'} \pi(s,\acute{a})Q(s,\acute{a}) > \bar{\pi}(s,\acute{a})Q(s,\acute{a}) \\ = \delta_l & \text{de otro modo} \end{cases} \tag{3.11}$$

En relación con el problema de coordinación en las tareas completamente cooperativas, los agentes en el algoritmo WoLF-PHC toman encuentra las conductas de los demás agentes, de esta manera no es usado un método de coordinación explícita, al tomar en cuenta las acciones de los demás agentes se tiene seguro que cualquier tipo de empate en dos o mas acciones igualmente óptimas sea rota por todos los agentes de la misma manera, de esta forma el algoritmo desarrolla una forma implícita de coordinación, donde los agentes aprenden a preferir la misma acción entre soluciones que son igualmente óptimas pasando por alto las demás alternativas [56] .

Por medio del anterior algoritmo, se construye una tabla con los valores óptimos de estado-acción para cada agente, en donde no fue necesario incorporar información a priori, salvo por la función de transición de estados y la función de recompensas propias del sistema . Después de cierto tiempo de interación los valores estado-acción convergen a un valor fijo, indicando que el proceso de aprendizaje ha finalizado. Por lo tanto el algoritmo WolF-PHC es capaz de encontrar el máximo valor Q de cada estado que ha sido visitado por los agentes.

Este método de aprendizaje por reforzamiento se basa en medidas exactas de señales que provienen del entorno tales como estado, acción tomada por los otros agentes, recompensa obtenida, de esta manera si la percepción del entorno por los agentes difieren en algún punto,

esto nos puede llevar a que cada agente actualice su tabla de valores estado-acción de manera distinta, y la consistencia de las tablas de valores Q no podría ser garantizada.

Otra dificultad que emerge del algoritmo WoLF-PHC es el problema de dimensionalidad, la cual se agrava conforme el numero de estados en el entorno, las acciones disponibles para los agentes y el mismo numero de agentes se incrementa, haciendo intratable problemas cuyo numero de estados es demasiado grande o contiene muchos agentes, afectando el tiempo de computo necesario para una adecuada convergencia de los valores buscados.

Por lo anterior buscamos mejorar el rendimiento del algoritmo WoLF-PHC por medio de añadir las útiles características de aproximación no lineal de las redes neuronales y del método kernel smoothing, lo anterior con la finalidad de superar los problemas de dimensionalidad y evitar el volver a ejecutar el algoritmo por completo cuando las características del entorno cambien o las condiciones iniciales de la tarea sea diferente.

3.2. Aprendizaje utilizando Kernel Smoothering

Los algoritmos que utilizan aprendizaje por reforzamiento obtienen una política de acciones óptimas a partir de los valores Q óptimos obtenidos durante el proceso de aprendizaje, la mayoría de estos métodos se basan en consideraciones discretas del entorno y en un numero limitado de estados,acciones y agentes con la finalidad de evitar el problema de la dimensionalidad. En virtud de que la mayoría de las aplicaciones reales tiene un gran numero de estados o incluso un numero infinito y que el algoritmo WoLF-PHC no trabaja de una manera eficiente con una gran cantidad de estos, es necesario proponer nuevas estrategias para volver mas tratables problemas donde existen una gran cantidad de datos disponibles.

Por lo anterior es usado el método de Kernel smoothing para aproximar los estados desconocidos que no son logrados visitar cuando el algoritmo de aprendizaje por reforzamiento es llevado acabo, también para realizar generalizaciones cuando el entorno ha sido modificado ligeramente,en ambos casos evitando la necesidad de volver a calcular las políticas óptimas .

En la fase de aproximación asumimos que tenemos una colección de datos s_t provenientes de la observación de los agentes, tomado en el intervalo $t \in [\tau_1, \tau_2]$, estos datos provienen

del modelo:

$$\hat{s}_{t+1} = \phi(s_t) + \epsilon$$

donde $\phi(s_t)$ es una curva de respuestas suave desconocida y ϵ es el error. El objetivo es encontrar una estimación de kernel $\hat{\phi}$ en algún punto de un tiempo pre especificado t. El kernel es simplemente una promediación ponderada de todos los puntos de datos:

$$\hat{\phi}(s_t) = N^{-1} \sum_{k=\tau_1}^{\tau_2} W_t s_t a(t) \tag{3.12}$$

donde $N = \tau_2 - \tau_1 + 1$, W_k es la secuencia de pesos. El estado estimado es denotado por : $\hat{s}_{t+1} \in R^n$.

Una representación conceptualmente simple de la secuencia de pesos W_t, es mediante la descripción de la forma de la función de pesos por medio de una función de densidad con un parámetro de escalamiento que tenga la función de ajustar el tamaño y las formas de los pesos cerca de los puntos de datos s_t. Es común referirnos a esta función conformadora como un kernel $K[z]$. El kernel es una función real, continua, acotada y simétrica el cual es integrable a uno.

$$\int K[z]\, dz = 1$$

La secuencia de pesos para el kernel smoother es definida por :

$$W_t s_t = K_t[s(t)] / \hat{f}_k(s_t)$$

donde $\hat{f}(s_t) = N^{-1} \sum_{k=\tau_1}^{\tau_2} K_k[s(t)]$, $K_k[s(t)]$ es una función gaussiana :

$$K_t[s(t)] = a \exp\left(\frac{\|s_t - \mathbf{c}_t\|^2}{2\sigma^2}\right), \quad a > 0 \tag{3.13}$$

La regresión llevada a cabo por el Kernel para los datos $[a(k), s(t)]$ de acuerdo a Nadaraya-Watson [57],donde $t \in [\tau_1, \tau_2]$ es:

$$\hat{\phi}(s(t)) = \frac{\sum_{t=\tau_1}^{\tau_2} K_k[s(t)]\, a(t)}{\sum_{t=\tau_1}^{\tau_2} K_k[s(t)]} \tag{3.14}$$

La mayoría de los análisis estadísticos para regresiones usando un kernel son difíciles, ya que son definidas como la razón de dos variables aleatorios $\hat{f}_t$ y K_t. En muchas importantes aplicaciones de procesamiento de señales y control automático una forma mas simple de regresión puede ser aplicada, en la cual es considerablemente mas fácil de realizar un análisis estadístico [57].

En las aplicaciones en control automático todas la variables aleatorias tiene una función de densidad de probabilidad constante, con este tipo de variables aleatorias, la sumatoria que implica la función kernel es equivalente a una integración de Montecarlo:

$$K_t\left[s\left(t\right)\right] \rightarrow \frac{N}{\tau_2}$$

Por lo tanto la regresión propuesta por Nadaraya usado en el kernel puede ser reemplazada por la regresión propuesta por Priestley-Chao [58], el cual es definida como:

$$\hat{\phi}\left(s\left(t\right)\right) = \frac{\tau_2}{N}\sum_{t=\tau_1}^{\tau_2} K_t\left[s\left(t\right)\right] a\left(t\right) \tag{3.15}$$

Dado lo anterior la expresión (3.14) es usado para modelar los estados desconocidos o que aun no han sido explorados del entorno. Una forma posible es diseñar una función kernel amplia con el fin de abarcar una mayor cantidad de datos por medio de una amplitud completa:

$$\sigma_{\text{máx}} = 2\sqrt{2\ln\left(2\right)}\sigma \tag{3.16}$$

donde $\sigma_{\text{máx}}$ es escogida en función del numero total de datos obtenidos,por ejemplo $\sigma_{\text{máx}} = \frac{N}{10}$, el parámetro de amplitud de la función gaussiana viene dado por :

$$\sigma = \frac{N}{20\sqrt{2\ln\left(2\right)}} \tag{3.17}$$

donde N es el total de datos disponibles. Los puntos medios de cada función Gaussiana $\left[x_j^*, y_j^*\right]$, $j = 1\cdots v$ son utilizados para construir las funciones de membresia Para el conjunto de datos $\left[a\left(t\right), s\left(t\right)\right]$, $k \in \left[1, N\right]$ se extraen reglas difusas en la forma de (3.14). x_j^* es el centro de la función gaussiana $\mu_{A_i^j}$, donde y_j^* es el centro de la función gaussiana μ_{B^j}.

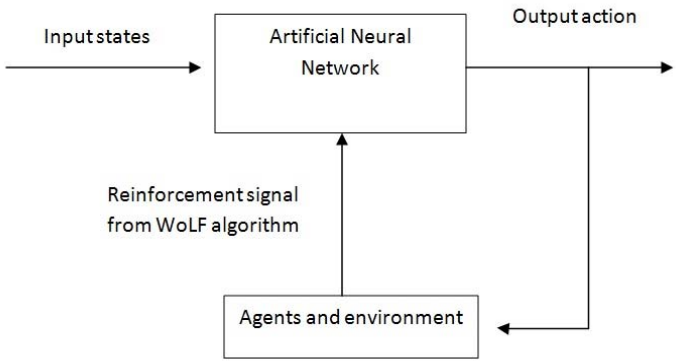

Figura 3.2: Procedimiento de entranamiento para la red neuronal

3.3. Aprendizaje utilizando redes neuronales

Las redes neuronales tienen una habilidad poderosa de fusión de datos y tolerancia a fallas. Ya que las redes neuronales de una sola capa solo resuelven los problemas de clasificación lineales, una red neuronal con múltiples capas es utilizada para estimar las acciones mejor valuadas que serán usadas por los agentes en los estados que no fueron visitados durante la fase de aprendizaje por reforzamiento. El flujo de aprendizaje es mostrado en la Figura 3.2

Las acciones disponibles para los agentes en los estados no visitados durante la fase de aprendizaje por reforzamiento serán aproximados por la siguiente red neuronal compuesta por 3 capas ocultas:

$$\hat{a}_{t+1} = W\sigma(V\,[s_t]) \tag{3.18}$$

donde $W \in R^{n\times m}$, $V \in R^{m\times n}$, m es el numero del nodo oculto el cual es diseñado por el usuario, n es el numero de agentes que se encuentran en el entorno, V representa los pesos en la capa oculta, σ es una función de activación neuronal, se usa una función sigmoidea en esta propuesta.

La función sigmoidea cuya forma es una s estilizada, es por mucho la forma mas común de

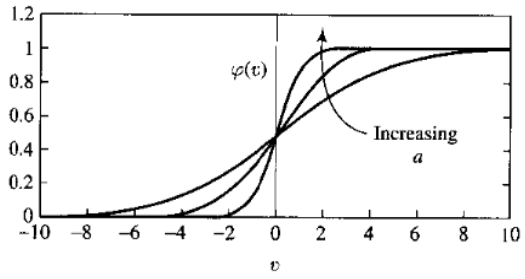

Figura 3.3: Funcion sigmoide para varios valores de pendiente a

activación usada en la construcción de redes neuronales artificiales. La cual es definida como función estrictamente creciente que exhibe un balance atractivo entre el comportamiento lineal y no lineal . Un ejemplo de una función sigmoidea es la función logística definida como :

$$\psi(v) = \frac{1}{1 + e^{(-av)}} \tag{3.19}$$

donde a es es el parámetro de pendiente de la función sigmoidea. Por medio de la variación del parámetro a se obtienen funciones sigmoideas de diferentes pendientes,como lo muestra la Figura 3.3.

En el limite, cuando la pendiente se aproxima a infinito la función sigmoidea se convierte en una simple función limite o umbral, definida como:

$$\varphi(v) = \begin{cases} 1 \text{ si } \ v \geq 0 \\ 0 \text{ si } v < 0 \end{cases} \tag{3.20}$$

Mientras una función umbral asume valores de 1 o 0 , una función sigmoidea asume un rango continuo de valores desde 0 a 1. Es de notar que la función sigmoidea es diferenciable (el cual es una importante característica en la teoría de las redes neuronales artificiales) mientras que la función umbral no lo es.

La función de activación definida en la expresión (3.19) varia su rango desde 0 hasta $+1$. En ocasiones es deseable tener funciones de activación que su rango sea desde -1 hasta

+1, en este caso la función de activación asume una forma anti-simétrica con respecto al origen, es decir, la función de activación es una función impar del campo inducido local, específicamente la función umbral definida por (3.20) ahora puede ser definida como

$$\varphi(v) = \begin{cases} 1 \text{ si } \ v > 0 \\ 0 \text{ si } v = 0 \\ -1 \text{ si } v < 0 \end{cases} \tag{3.21}$$

lo cual es referida como una función signo. Por la forma correspondiente de una función sigmoidea se puede utilizar una función tangente hiperbólica definida por

$$\varphi(v) = \tanh(v) \tag{3.22}$$

Al permitir una función de activación de la forma sigmoidea que asuma valores negativos como en la expresión (3.22) se obtienen beneficios analíticos en las redes neuronales artificiales [54].

Las entradas para la red neuronal son los estado s_t y la salida de la red serán las acciones aproximadas $\hat{a}_t$ que los agentes deberán de tomar para llevar a cabo la meta encomendada. Es de notar que el aprendizaje utilizado en las redes neuronales es un método de aprendizaje supervisado. Por lo que las muestras para el entrenamiento de la red neuronal provendrán de la fase anterior de aprendizaje, donde se utiliza el algoritmo de aprendizaje por reforzamiento WoLF-PHC, el cual nos brindará como entradas de entrenamiento los estados del sistema y como salida las acciones óptimas encontradas para cada uno de los agentes.

El error entrenamiento $e(k)$ a la salida de la neurona j esta definido como

$$e_j(k) = y_j(k) - d_j(k)$$

donde $y_j(k)$ es el estado sensado por el agente, y $d_j(k)$ es el patrón de acciones de los agentes. Las sumas instantáneas de los errores al cuadrado de la salida es dado por

$$\varepsilon(k) = \frac{1}{2}\sum_{j=1}^{l} e_j^2(k)$$

donde l es el numero de neuronas de la capa de salida. Usando el algoritmo back propagation para entrenar la red neuronal , el peso que conecta la neurona i con la neurona j es actualizado de la siguiente manera:

$$w_{ij}(k+1) = w_{ij}(k) - \eta \frac{\partial \varepsilon(k)}{\partial w_{ij}(k)}$$

donde η es el parámetro de razón de aprendizaje del algoritmo back propagation. El termino $\frac{\partial \varepsilon(k)}{\partial w_{ij}(k)}$ puede ser calculado:

$$\frac{\partial \varepsilon(k)}{\partial w_{ij}(k)} = \frac{\partial \varepsilon(k)}{\partial e_j(k)} \frac{\partial e_j(k)}{\partial y_j(k)} \frac{\partial y_j(k)}{\partial v_j(k)} \frac{\partial v_j(k)}{\partial w_{ij}(k)}$$

Las derivadas parciales están dadas por

$$\frac{\partial \varepsilon(k)}{\partial e_j(k)} = e_j(k), \frac{\partial e_j(k)}{\partial y_j(k)} = 1, \frac{\partial y_j(k)}{\partial v_j(k)} = \varphi_j'\left[v_j(k)\right], \frac{\partial v_j(k)}{\partial w_{ji}(k)} = y_i(k)$$

Una función lineal es utilizada en la capa de salida de la red neuronal

$$w_{ij}(k+1) = w_{ij}(k) - \eta y_i(k) e_j(k) \tag{3.23}$$

De tal manea que w_{ki} puede ser actualizado como:

$$w_{ki}(k+1) = w_{ki}(k) - \eta \frac{\partial \varepsilon(k)}{\partial w_{ki}(k)}$$

además

$$\begin{aligned} &\frac{\partial \varepsilon(k)}{\partial w_{ki}(k)} = \frac{\partial \varepsilon(k)}{\partial e_j(k)} \frac{\partial e_j(k)}{\partial y_j(k)} \frac{\partial y_j(k)}{\partial v_j(k)} \frac{\partial v_j(k)}{\partial y_i(k)} \frac{\partial y_i(k)}{\partial v_i(k)} \frac{\partial v_i(k)}{\partial w_{ki}(k)} \\ &= \left\{ e_j(k) \varphi_j'\left[v_j(k)\right] W_{ij} \right\} \varphi_i'\left[v_i(k)\right] y_k(k) = e_i(k) \varphi_i'\left[v_i(k)\right] y_k(k) \end{aligned}$$

Por lo tanto

$$w_{ki}(k+1) = w_{ki}(k) - \eta y_k(k) e_i(k) \varphi_i'\left[v_i(k)\right] \tag{3.24}$$

donde $e_i(k) \varphi_i'\left[v_i(k)\right]$ es el error en back propagation.

3.4. Planeación de Trayectoria

En cada paso de tiempo discreto t , los estados donde se encuentran los agentes son observados y los estos son referidos a una tabla de estados-acciones llamada Q-tabla. En este capitulo usamos el algoritmo de aprendizaje por reforzamiento para sistemas multi- agentes llamado WoLF-PHC con el objetivo de obtener acciones óptimas, a partir de los datos de Q^* que se obtiene al final de la convergencia del algoritmo.

Cuando los estados en donde se encuentran los agentes no estén disponibles en la Q-tabla, ya sea por que estos no fueron explorados durante el algoritmo de aprendizaje por reforzamiento o por la ocurrencia de pequeños cambios en el entorno, las acciones a realizar por los agentes serán aproximadas ya sean por el kernel smoother (3.12) o por la red neuronal segun sea el caso elegido. La nueva aproximación sera enviada a los agentes con la finalidad de continuar la tarea encomendada.

Esta integración entre la $Q-$tabla y los métodos de aproximación no lineales pueden mejorar el desempeño general del proceso de aprendizaje que se lleve a cabo en entornos dinámicos o que no han sido explorados en su totalidad, cuando un estado desconocido sea encontrado por los agentes esta información relativa al estado será alimentada al aproximador.

El aproximador previamente entrenado genera como salida una acción suboptima para cada agente en el entorno, de esta manera se ahorra tiempo de computo al evitar volver ejecutar el algoritmo de aprendizaje por reforzamiento cuando los agentes enfrentan un estado no conocido.

El método propuesto puede ser enumerado en los siguientes pasos:

1. **Se captura los estados iniciales de cada ciclo de entrenamiento del algoritmo de aprendizaje por reforzamiento WoLF-PHC** : el estado actual del estado de los agentes con respeto al entorno son capturados a través de sensores.

2. **Limitar la cantidad de estados capturados**: La limitación de los estados capturados reduce el conjunto de estados que los agentes requieren para completar la tarea lo que permite ahorrar tiempo y poder de computo.

3. **Establecer las acciones disponibles para los agentes** : En cada momento los agentes son requeridos de realizar una acción con un grado de coordinación, por lo tanto es necesario seleccionar de antemano las acciones mas fiables a realizar por los agentes, con el objetivo de mantener minimizado el espacio de acciones y evitar problemas de dimensionalidad.

4. **Estimar los valores Q estado-acción de cada agente**: La recompensa numérica de cada acción es calculada y dada al agente después de que se realiza una acción en conjunto, este calculo se realiza por medio del algoritmo de aprendizaje por reforzamiento Win or Learn Fast -Policy Hill Climbing (WoLF-PHC), los valores obtenidos son guardados en una tabla de busqueda llamada $Q-$tabla.

5. **Repetir los pasos 2-4 hasta que los agentes alcance el objetivo planteado**: El ciclo de entrenamiento finaliza bajo los siguientes supuestos: si el recorrido de aprendizaje de los agentes alcanza un máximo numero de iteraciones, si algún agente colisiona con otro agente o algún obstáculo del entorno o si los agentes alcanzan a cumplir el objetivo final planteado.

6. **Repetir los pasos 2-5 hasta que los valores Q converjan** : Esto sucede cuando los valores Q permanecen sin cambios o ellos se encuentran debajo de alguna cota establecida de antemano.

7. **Obtención de Q-table final de estados-acciones**: La tabla final de estados-acciones óptimas es puesta a punto para la selección de las acciones óptimas por medio de la localización de la acción que generara el máximo valor Q en cada estado.

8. **Fase de entrenamiento de la red neuronal** : Se usa la tabla de valores Q obtenida por el algoritmo WoLF-PHC para entrenar la red neuronal, cada columna de la tabla Q representa un estado, la cual es introducido en la red neuronal como entrada y las acciones óptimas encontradas como salidas del sistema.

9. **Obtención de las acciones óptimas aproximadas:** Una vez entrenada la red neuronal, esta proveerá una acción aproximada conjunta que implementarán los agentes cuando estén encarando estados desconocidos que no hubieran sido explorados ni aprendidos en las etapas anteriores de aprendizaje.

3.5. Resultados en simulación

El principal objetivo de las simulaciones ejecutadas en MATLAB es revisar, validar la operación y ejecución del método propuesto, además de comparar los resultados obtenidos antes de una implementarlos en experimentos en tiempo real usando los robots móviles Khepera [59]. El ejercicio de simulación es una tarea cooperativa para un sistema multiagentes integrado por 2 entidades, donde es necesario emplear coordinación entre ellos.

El entorno en donde se desarrollara la actividad es divida en una rejilla de 7×8 celdas, en donde los agentes necesitan transportar un objetivo (de color rojo) hacia una base que esta localizada en la parte superior en un tiempo mínimo debiendo evitar obstáculos y coordinarse entre ellos al momento pasar por la sección estrecha del camino.

La Figura 3.4 muestra las posiciones iniciales de los dos agentes, en cada paso de tiempo, los agentes se pueden mover una celda, si la celda a la cual se quieren mover no esta vacía ellos no podrán ocupar ese lugar. El primer objetivo a cumplir es que ambos agentes tomen a cada lado el objeto a mover, si los dos agentes tienen tomado el objeto, ellos se deben de desplazar en forma conjunta a la base final con la idea de depositar el objetivo rojo, como lo muestra la Figura 3.5.

En la presente simulación es asumido que los agentes no conocen a priori la posición de los obstáculos del entorno, pero la posición inicial de los agentes , la posición del objetivo que deben de tomar los agentes y la posición final de la base si son conocidos por los agentes.

Las variables de estado para posición para cada agente i son las siguientes:

$$\begin{aligned} x &\in (1, 2, 3, .., 8) \\ y &\in (1, 2, 3, .., 7) \end{aligned}$$

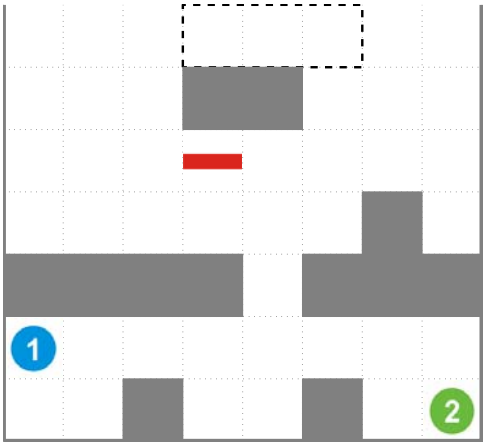

Figura 3.4: Posiciones iniciales de los agentes

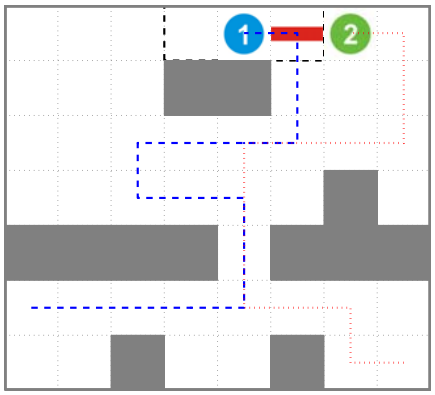

Figura 3.5: Trayectoria optima despues de la fase de aprendizaje por reforzamiento WoLF-PHC

Cuadro 3.1: The parameters of WoLF-PHC algorithm used in the simulation task.

Parametro de exploracion ε	0,7
Factor exponencial de decaimiento λ	0,5
Parametro WoLF-PHC	$\delta_{win} = 0{,}1$
Parametro WoLF-PHC	$\delta_{lose} = 0{,}35$
Razon de aprendizaje α	0,1
Factor de descuento γ	0,98

La variable de estado que muestra la relación entre los agentes y el objetivo a tomar :

$$g \in (left, right, free)$$

Por lo que el vector de estado para esta tarea es :

$$s = [x_1, y_1, g_1, x_2, y_2, g_2]$$

donde la cardinalidad de S es

$$|S| = (8 \cdot 7 \cdot 3)^2 = 112896$$

Las acciones disponibles para los agentes viene dado por :

$$A = \{\text{izquierda,derecha,arriba,abajo,estar quiero}\}$$

La Q-tabla obtenida después de ejecutar el algoritmo WoLF-PHC y obtener los valores óptimos esta integrada por $|S| \times 5 = 564480$ elementos, los parámetros utilizados al ejecutar este algoritmo vienen dados en la tabla (3.1).

Después de 24 ensayos de entrenamiento los valores de la Q-tabla convergen a los valores óptimos, como lo muestra la Figura 3.6, la coordinación entre los agentes se llevo a cabo

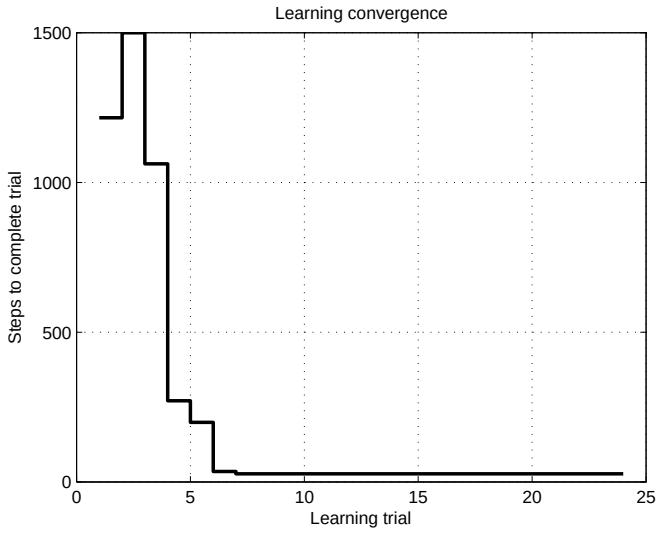

Figura 3.6: Convergencia en la primera fase de aprendizaje.

mediante una forma de coordinación implícita, donde los mismos agentes aprenden a preferir una acción óptima mientras desestiman las demás alternativas presentes.

La fase de entrenamiento de la red neuronal y el kernel se presenta después de haber obtenido la Q-tabla de valores óptimos, estos valores serán utilizados como muestras de entrenamiento para la red neuronal artificial y el kernel. La red neuronal es entrenada por 23 épocas siendo 1500 el numero máximo de iteraciones permitidas.

Las entradas al sistema son $s = [x_1, y_1, g_1, x_2, y_2, g_2]$ y la salidas aproximadas serán las acciones que deberán implementar los agentes, estas acciones serán movimientos a la : 1-izquierda, 2-derecha,3-hacia abajo, 4- hacia arriba, 5-no moverse. En el caso de que la salida aproximada sea un numero no entero este se procederá redondear al numero inmediato superior a partir de 0,6. Los parámetros utilizados en el entrenamiento de la red neuronal es dado en la tabla (3.2).

Después de finalizar el entrenamiento de la red neuronal y del kernel, procedemos a validar nuestra propuesta por medio de iniciar la tarea colocando a los agentes en estados no visitados, es decir, estados que no se encontraban originalmente en la Q-tabla obtenida

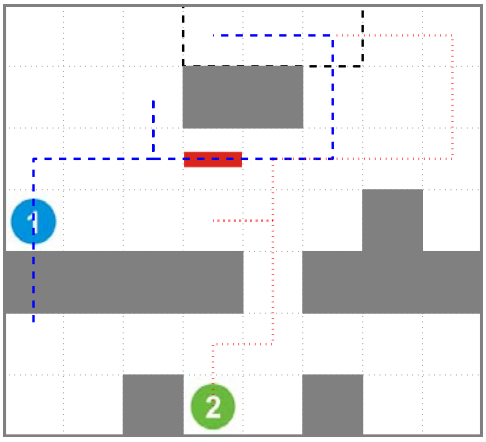

Figura 3.7: Trayectoria final encontrada en simulacion por medio de la red neuronal partiendo desde un estado desconocido

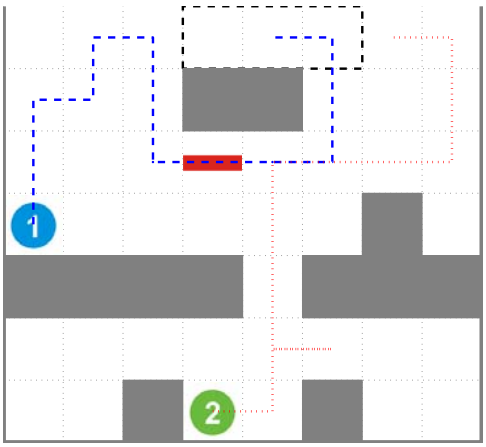

Figura 3.8: Trayectoria final encontrada en simulacion por medio del Kernel aproximador partiendo desde un estado desconocido

Cuadro 3.2: Parametros usados en el entrenamiento de la red neuronal (Back-Propagation)

Numero de capas ocultas	3
Numero de neuronas en la 1^{ra} capa oculta	50
Numero de neuronas en la 2^{da} capa oculta	20
Numero de neuronas en la 3^{ra} capa oculta	10
Funcion de activacion en la capa de salida	Linear function
Funcion de activacion en capas ocultas	Sigmoid function

mediante el algoritmo de aprendizaje por reforzamiento, se observa que en la mayoría de los estados desconocidos los agentes llegan a la posición del objetivo satisfactoriamente y posteriormente lo transportaron a la base final . Ambos agentes no toman el camino mas corto desde estas posiciones iniciales desconocidas, posiblemente debido a errores de aproximación de la red neuronal y del kernel. El desempeño de los agentes pueden observarse en las Figuras 3.8 y 3.7.

3.6. Resultados Experimentales

Con la finalidad de validar el desempeño del método propuesto, se usa dos robots móviles Khepera en una aplicación experimental, cuyo objetivo es el de completar la misma tarea cooperativa conducida en las simulaciones de la sección anterior, se asume que los agentes no tiene un conocimiento previo acerca de la posición y forma de los obstáculos presentes en el entorno.

La posición inicial de los agentes se selecciona aleatoriamente y se lleva a cabo 2000 pasos de aprendizaje, en el caso de alcanzar este limite, el experimento es detenido y vuelto a reiniciar. En el momento que los agentes encuentren la trayectoria óptima sin colisionar con los obstáculos u otros agentes se dirá que los valores de la Q-tabla han convergido, por

Cuadro 3.3: Parametros del entorno usados en la experimentacion

Tamaño del entorno	3000 x 1800 mm
Puntos iniciales de la 1^{ra} prueba	agente 1=$(337, 115)$ agente 2=$(1462, 115)$
Puntos iniciales de la 2^{da} prueba	agente 1=$(112, 1500)$ agente 2=$(514, 115)$
Puntos iniciales de la 3^{ra} prueba	agente 1=$(337, 115)$ agente 2= $(562, 1500)$
Posicion del objetivo	$x \in [675, 900]$ $y \in [2307, 2538]$
Posicion de la base	$x \in [450, 1350]$ $y \in [2760, 3000]$

lo que se detendrá la etapa de aprendizaje por reforzamiento. La configuración del entorno utilizado es mostrado en la Figura 3.10

Los pares estados-acciones óptimas encontrados en la Q-tabla serán introducidos en la red neuronal y en el Kernel como muestras de entrenamiento, de esta manera el controlador mixto sera capaz de aproximar la mejor acción para cada agente cuando estos se encuentre en estados desconocidos. Los parámetros utilizados en experimento son mostrados en la tabla (3.3)

Los dispositivos Khepera son robots móviles con ruedas de pequeño tamaño los cuales cuentan con una serie de características apropiadas para el desarrollo de sistemas multi-agentes. Cada robot Khepera tiene 5 sensores los cuales están localizados alrededor del robot los cuales son posicionados y numerados como lo muestra la Figura 3.9.

Los 5 sensores son 5 pares de dispositivos ultrasónicos donde cada par es compuesto de un transmisor y un receptor, estos sensores ultrasónicos son usados para reconstruir la odometría y detectar las características del entorno físico. En la etapa inicial de aprendizaje los estados son capturados, el estado S_t corresponde a un vector de 16 elementos (8 elementos para cada agente), este estado S_t es representado por :

$$S_t = [l_{a,c}, d_a, p_a, g_a]$$

Donde $a = 1, 2$ y $c = 1, .., 5$, los elementos $l_{a,c}$ son los 5 sensores ultrasonicos de cada robot

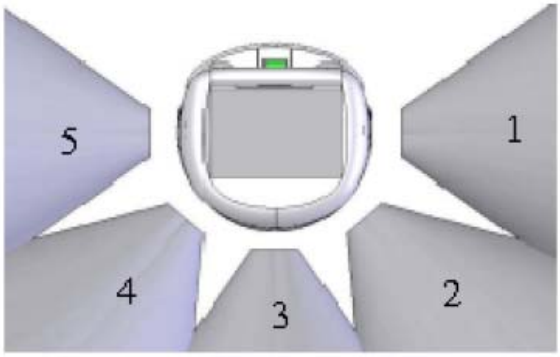

Figura 3.9: Posicion de los 5 sensores ultrasonicos

Khepera, los cuales indican las posiciones de los obstáculos y los otros agentes presente en el entorno. El elemento d_a representa la distancia relativa entre cada robot Khepera y el objetivo, después de que el objetivo ha sido tomado d_a representa la distancia entre los agentes y la base, el elemento p_a representa el ángulo relativo entre el eje de movimiento hacia adelante de cada robot y el objetivo, después que el objetivo ha sido tomado p_a representa el ángulo relativo entre el eje de movimiento hacia adelante de cada robot y la base, finalmente el elemento g_a es una variable la cual indica el lado en el cual el agente ha tomado el objetivo, la cual toma los valores:

$$g_a \epsilon \{\text{libre,tomado-izquierda,tomado-derecha}\}$$

La variable g_a es necesaria para asegurar que el vector de estado mantiene la propiedad de Markov, donde un proceso estocastico tiene la propiedad de Markov si la distribución de probabilidad de los estados futuros del proceso solo depende del estado presente y no de la secuencia de eventos que le precedieron.

Con la idea de reducir el numero total de estados que los agentes necesitan para completar la tarea encomendada, el tamaño del espacio es reducido a través de la cuantificación de los estados presentes, ya que usando un espacio de aprendizaje muy grande tomara demasiado tiempo completar, generando un problema de dimensionalidad, por otro lado si se usa un

Figura 3.10: Configuracion Experimental

espacio de aprendizaje muy pequeño este no sera suficiente para representar el entorno de manera adecuada [60].

Las 5 lecturas de los sensores ultrasónicos $l_{a,c}$ son cuantificados en tres grados representando la cercanía hacia el obstáculo mas cercano o los demás agentes, 0 representa que los obstáculos o agentes se encuentran cercanos, 1 indican que los obstáculos o agentes están a una distancia media y finalmente 2 representa que los obstáculos o agentes están relativamente lejos de los sensores.

Los parámetros d_a (la distancia hacia el objetivo o la base) y p_a (el ángulo relativo hacia el objetivo o la base) son divididos en 8 grados $(1, 2, 3, ..., 8)$ donde 1 representa la distancia o ángulo mas pequeño y el numero 8 representa la distancia o ángulo mas grande desde la posición del agente hasta el objetivo o la base final.

Con la finalidad de completar la tarea cooperativa encomendada, es requerido que cada agente escoja una acción de las 4 acciones disponibles

- Mover hacia adelante 5 cm
- Dar vuelta 25° en la dirección del reloj
- Dar vuelta 25° en la dirección contraria al reloj
- No moverse

Los sensores ultrasónicos de los robots Khepera son utilizados para ayudar a los robots móviles a determinar si hay algún obstáculo cercano en el entorno, mientras estos se mantienen navegando hacia el objetivo final. Los experimentos llevados a cabo revelan que el algoritmo de aprendizaje por reforzamiento confía fuertemente en las lectura de los sensores, estas lecturas en una situación de simulación ideal , se basan en cálculos matemáticos los cuales son exactos y consistentes.

En la implementación experimental las lecturas desde los sensores ultrasónicos son inexactas y fluctuantes, por lo que es necesario evitar que estas lecturas imprecisas no afecten la fase de aprendizaje, lo que podría generar información que describiera a los estados de manera poco precisa, lo que nos llevaría a una falsa convergencia de los datos en la Q-tabla . Este efecto es minimizado por medio de permitir un periodo de espera después que sea realice una acción conjunta por parte de los agentes, asegurando que los sensores tienen lecturas en estado estable antes de proceder a registrarlo como parte del aprendizaje.

Además al segurarnos que el movimiento de los agentes se realiza a una velocidad baja durante el proceso de aprendizaje, las colisiones con obstáculos u otros agentes pueden ser reducidas. Por ultimo es de notar, que la cuantificación de los estados debe de ser suficiente para representar la ubicación actual cuando los agentes se encuentran en movimiento.

La función de recompensa para la tarea completamente cooperativa que se presenta en esta sección esta dada :

$$
\begin{aligned}
r_{k+1} &= 1 \text{ si el agente toma el objetivo} \\
r_{k+1} &= 10 \text{ si los agentes llevan el objetivo hasta la posición base} \\
r_{k+1} &= 0 \text{ en cualquier otra situación}
\end{aligned}
$$

Cuadro 3.4: Parametros utililzados en el algoritmo WoLF-PHC parte experimental

Parametro de exploracion ε	0,8
Factor exponencial de decaimiento λ	0,7
Parametro WoLF-PHC	$\delta_{win} = 0{,}25$
Parametro WoLF-PHC	$\delta_{lose} = 0{,}45$
Razon de aprendizaje α	0,15
Factor de descuento γ	0,96

La recompensa numérica en cada par estado-acción es dada a los agentes inmediatamente que la acciones realizada por ellos.

La estimación de los valores en la Q-tabla para cada estado-acción de cada agente se realiza a través del algoritmo de aprendizaje por reforzamiento WoLF-PHC [3], los parámetros utilizados son mostrados en la tabla (3.4), en esta etapa los agentes perciben los estados del entorno y aprenden las acciones óptimas a realizar.

El algoritmo de aprendizaje por reforzamiento evalúa las acciones de los agentes que estos han realizado a partir de una señal informativa en forma de recompensa o penalidad [20]. Cada ciclo de entrenamiento finalizara si un robot colisiona con algún obstáculo o con algún otro agente, en tal caso el ciclo es reiniciado. El proceso de aprendizaje continuara hasta que los valores de la Q-tabla se mantengan sin cambios, lo que implica que los pares de datos estado-acción sean óptimos, la Figura 3.11 muestra la trayectoria óptima desde una posición inicial arbitraria previamente aprendida. La convergencia del algoritmo WoLF-PHC es mostrada en la Figura 3.12.

Las acciones óptimas que los agentes ejecutaran en cada estado S_t son obtenidas al localizar el máximo Q-valor para el estado correspondiente en el que se encuentran:

$$\pi^* = \arg \max_a Q^* (S, A) \tag{3.25}$$

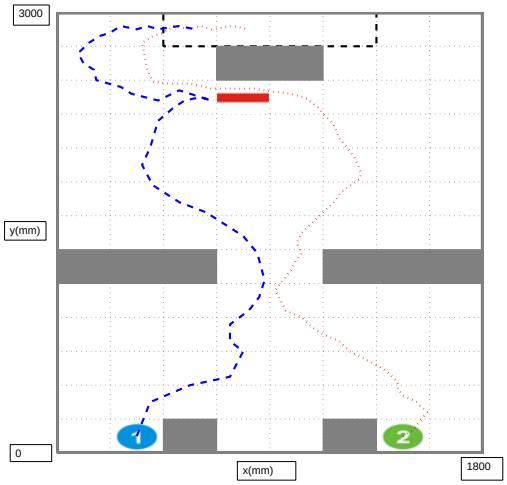

Figura 3.11: Trayectoria experimental generada por el algoritmo WoLF-PHC

El conjunto de datos que serán empleados como muestras de entrenamiento para la red neuronal y al Kernel son tomados de la Q-tabla óptima generada por el aprendizaje por reforzamiento, cada columna de la Q-tabla representa un estado, y la salida del aproximador sera la acción conjunta que los agentes deberán ejecutar. El tipo de red neuronal artificial sera del tipo feedward back propagation, se usa el neural network toolbox proveido por MATLAB [62]. La adición de la red neuronal y del Kernel tienen la aspiración de mejorar el controlador descentralizado, donde los parámetros utilizados para la red neuronal se muestran en la tabla (3.5).

Con la finalidad de comparar el desempeño entre el método de aproximación usando Kernel y las redes neuronales artificiales, se escoge como posición inicial para los agentes, un estado desconocido el cual no se encuentra en la Q-tabla generada por el algoritmo WoLF-PHC, bajo esta situación no seria posible proveer las acciones óptimas a los agentes, por lo que el kernel o la red nueronal ofrecerá una acción suboptima coordinada para cada agente, los aproximadores aprenden estas acciones al usar como muestras de entrenamiento los datos obtenidos en al fase previa de aprendizaje, ejemplo de estas situaciones se muestran en las

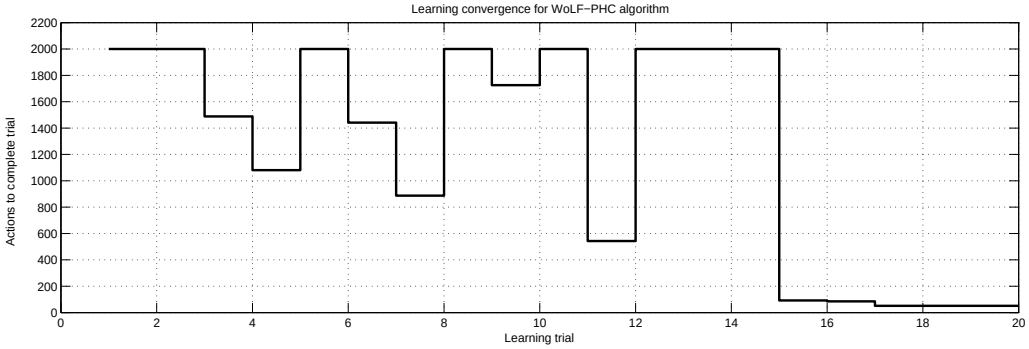

Figura 3.12: Grafica de convergencia algoritmo WoLF-PHC

Figuras 3.13 y 3.14 , donde los agentes tienen posición inicial en estados que no se encuentran en la Q-tabla.

Con los resultados obtenidos en simulación y en la implementación experimental con los robots móviles Khepera, podemos concluir que el método propuesto, por medio de la combinación de aprendizaje por reforzamiento para multi-agentes con métodos de aproximación no lineal ,confirman la confiabilidad y robustez del controlador para la planeación de trayectoria cuando los agentes enfrentan estados desconocidos, superando la necesidad de volver a ejecutar el algoritmo de aprendizaje por reforzamiento ahorrando tiempo y poder computacional.

Es necesario enfatizar que el método propuesto utiliza un modelo de estados discretos del sistema, esto es posible debido a la cuantización de los estados en el entorno, por lo que una natural dirección en el estudio de este tema seria buscar técnicas donde se pudiera lidiar con sistemas multi-agentes con estados continuos.

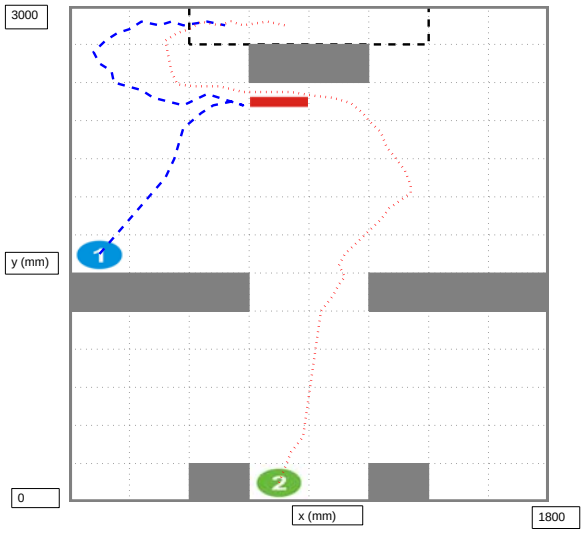

Figura 3.13: Trayectoria encontrada por el Kernel partiendo desde un estado desconocido.

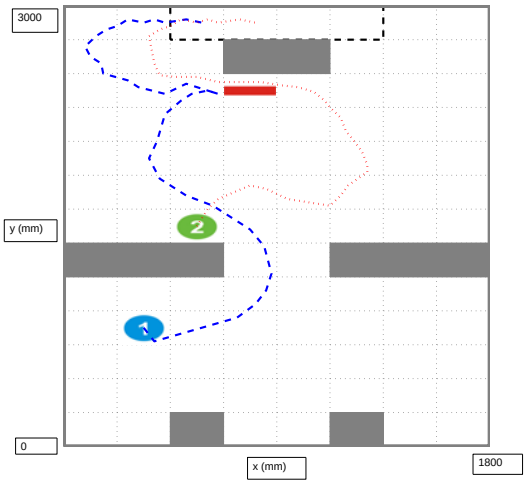

Figura 3.14: Trayectoria encontrada por la red neuronal partiendo desde un estado desconocido.

Cuadro 3.5: The parameters of the neural network back propagation used in the real time implementation.

Numero de capas ocultas	4
Numero de neuronas de la 1^{ra} capa oculta	40
Numero de neuronas de la 2^{da} capa oculta	25
Numero de neuronas de la 3^{ra} capa oculta	15
Numero de neuronas de la 4^{ta} capa oculta	10
Funcion de activacion en la capa de salida	Funcion Lineal
Funcion de activacion en capas ocultas	Funcion Sigmoide

Capítulo 4

Planeación de multi-agentes vía aprendizaje por reforzamiento difuso

El problema de la dimensionalidad mencionado en los capítulos anteriores, es una dificultad constante en los sistemas multi-agentes que aprenden por medio de reforzamiento, ya que el numero de estados presentes crecen exponencialmente conforme el tamaño del entorno y el numero de agentes en el entorno se incrementan..

La gran mayoría de los enfoques en aprendizaje por reforzamiento requieren de una representación exacta de los valores estado-acción y de las políticas de los agentes, los cuales son descritos en forma de tablas de búsqueda, de esta manera, el análisis de la tarea a realizar se puede complicar o incluso volver intratable, por lo anterior, las técnicas clásicas por reforzamiento se reduce a pequeñas tareas con estados discretos.

En las aplicaciones físicas, las variables de estado que describen el sistema cuentan con un gran numero de posibles valores o incluso podrían contar con valores continuos, por lo tanto las funciones de valor (las cuales le dará al agente la oportunidad de escoger la mejor acción disponible), deben de ser aproximadas de la manera mas adecuada, buscando minimizar el error en la aproximación y fomentando la coordinación entre los agentes con la finalidad de obtener una conducta conjunta coherente cuando estos realicen labores cooperativas. [63].

Algunos algoritmos han sido propuestos para lidiar con el problema de la dimensionalidad,

ya sea usando redes neuronales artificiales, con la idea de proporcionar acciones sub-óptimas a partir de una Q-tabla óptima [64]; otras utilizando funciones de aproximación para espacios grandes de acción-estados discretos [65]; algunas aplicando Kernel-Smoothers para la generalización de acciones [66]; otras propuestas usan cuantización vectorial para estados y acciones continuas [67]; las mas populares usando Q-learning y funciones normalizadas Gausianas como aproximadores [68]; entre otras .

En este capitulo se propone un enfoque para solucionar la dimensionalidad en los sistemas multi-agentes bajo una tarea cooperativa, mediante el uso de un algoritmo aprendizaje por reforzamiento a través de una cuantificación difusa del espacio de estados, combinado con una versión modificada del algoritmo WoLF-PHC [3], el aproximador considerado representara las funciones Q de cada uno de los agentes, este algoritmo propuesto utiliza una Q-iteration difusa basado en un modelo del sistema y la coordinación es realizada por medio de una coordinación implícita donde los agentes aprenden a seleccionar la acción óptima sobre las otras opciones disponibles.

4.1. Aprendizaje por reforzamiento mediante Q iteración

Generalmente en los sistemas multi-agentes se asume que existen un cierto numero de agentes probablemente heterogeneos (con distintas características de apredizaje), los cuales cuentan con un conjunto individual de acciones con la intención de resolver un problema en un entorno determinado.

Tal problema puede ser modelado por medio de un juego estocástico, que puede ser visto como una generalización de un proceso de decisión de Markov en el cual la acción ejecutada en cualquier estado consiste en las acciones individuales realizada por cada agente [7].

El juego estocástico deterministico es una tupla $(X, U_1, U_2, ..., U_n, f, \rho_1, \rho_2, ..., \rho_n)$ donde n es el numero de agentes en el entorno, X es el conjunto finito de los estados en el entorno, U_i $i = 1, 2, .., n$ son los conjuntos finitos de acciones disponibles para cada agente, el cual genera una serie de acciones conjuntas $\mathbf{U} = U_1 \times U_2 \times ... \times U_n$. Donde la función de transición de estados esta dada por $f : X \times U \rightarrow X$ y las funciones de recompensa $\rho_i : X \times U \rightarrow R$

$i = 1, 2, .., n$.

Como resultado de las acciones conjuntas de todos los agentes $\mathbf{u}_k = \left[u_{1,k}^T, u_{2,k}^T, ..., u_{n,k}^T\right]^T, \mathbf{u}_k \in \mathbf{U}$ aplicado en el estado x_k , los estados cambian a $x_{k+1} = f(x_k, \mathbf{u}_k)$, una función escalar de recompensa $r_{i,k+1} = \rho\left(x_k, \mathbf{u}_k\right)$ es dado para cada agente, el cual evalúa la efectividad de la acción conjunta $\mathbf{u}_k$. Las acciones son escogidas de acuerdo a su propia política de acciones de cada agente $h_i : X \times U_i \rightarrow [0, 1]$, donde el conjunto de todas las políticas individuales forman una política conjunta $\mathbf{h}$.

En vista del hecho de que las recompensas $r_{i,k+1}$ dependen de las acciones conjuntas, los retornos obtenidos por los agentes dependen de una política conjunta:

$$R_i^{\mathbf{h}}(x) = \sum_{k=0}^{\infty} \gamma^k \ r_{i,k+1}$$

de tal manera que las funciones Q de cada agente están en función de una política de acciones conjuntas con $Q_i^h = X \times \mathbf{U} \rightarrow \mathbf{R}$

$$Q_i^h\left(x, u\right) = E\left[\sum_{k=0}^{\infty} \gamma^k \ r_{i,k+1} \left| x_0 = x, \mathbf{u}_0 = \mathbf{u}, \mathbf{h}\right.\right]$$

En general cada agente puede tener sus propios objetivos a cumplir, en esta sección asumimos que la tarea a desarrollar es completamente cooperativa, de esta manera, las funciones de recompensa son las mismas para todos los agentes $\rho_1 = \rho_2 = ... = \rho_n$, y por lo tanto los retornos también serán los mismos $R_1^{\mathbf{h}} = R_2^{\mathbf{h}} = ... = R_n^{\mathbf{h}}$, de esta manera los agentes tendrán el mismo objetivo final, el cual sera maximizar el termino común de desempeño a largo plazo llamado retorno $R^{\mathbf{h}}$.

Determinar una política conjunta de acciones óptima $\mathbf{h}^*$ en un sistema multi-agente es un problema de selección de equilibrio [69]. Establecer un equilibrio en sistema multi-agentes es en general un problema complicado, sin embargo, la estructura de la tarea completamente cooperativa hace mas manejable esta situación, en esta sección se asume que todos los agentes conocen la estructura del juego en la forma la función de transición de estados f y las funciones de recompensa ρ_i, por lo anterior el algoritmo propuesto se basa en un modelo del entorno.

En un juego estocastico completamente cooperativo donde el objetivo de aprendizaje es maximizar un retorno R común, el objetivo puede ser alcanzado por medio de aprender los valores Q^* óptimos mediante iteración

$$Q(x_k, \mathbf{u}_k) = \rho\left(x_k, \mathbf{u}_k\right) + \gamma \max_j Q\left(f(x_k, \mathbf{u}_k), \mathbf{u}_j\right)$$

Los agentes podrán usar políticas codiciosas (greedy) para maximizar el retorno común [70] :

$$h_i^*(x) = \arg\max_{u_i} \max_{u_1, u_2, .., u_n} Q^*(x, \mathbf{u})$$

En algunos estados, múltiples acciones conjuntas pueden ser óptimas, en la falta de mecanismos de coordinación, los agentes podrían romper este tipo de empates de distintas maneras y alejarse de una acción conjunta óptima.

Sea el conjunto de las Q-funciones indicado por $\mathbf{Q}$. El mapeo presente en la Q-iteracion puede ser definida en el caso deterministico por medio:

$$H(q)(x, \mathbf{u}) = \rho\left(x, \mathbf{u}\right) + \gamma \max_j Q\left(f(x, \mathbf{u}), \mathbf{u}_j\right)$$

De acuerdo con Bertsekas [71], la Q función óptima Q^* satisface la ecuación de optimalidad de Bellman $Q^* = H(Q^*)$, por lo que Q^* es un punto fijo de $\dot{H}$. De tal manera podemos iniciar a partir de un valor arbitrario Q_0 y actualizar el valor de Q en cada iteración l por medio:

$$Q_{l+1} = H(Q_l) \tag{4.1}$$

En Istratesku [72], se demuestra que H es una contracción con factor $\alpha < 1$ en la norma infinita, por lo que cualquier par de $Q-$función Q_1 y Q_2 mantiene la relación:

$$\left\| H(Q_1) - H(Q_{2)} \right\| \leq \alpha \left\| Q_1 - Q_2 \right\|$$

H tiene un único punto fijo, por lo que Q^* es un punto fijo de :

$$H : Q^* = H(Q^*)$$

Por lo anterior la $Q-$iteración converge a Q^* conforme $l \to \infty$, de esta manera una política óptima $h_i^*(x)$ puede ser calculada desde Q^* usando la expresión (4.1), como se menciono anteriormente, para realizar la interación anterior es necesario el conocimiento de la tarea a realizar por medio de la función de transición de estados f y de la función de recompensa ρ_i.

Esta clase de métodos tales como $Q-$iteración están en la necesidad de guardar y actualizar distintos valores Q para cada par estado-acción, por lo que solo problemas con un conjunto finito de estados y acciones son generalmente tratados [73], [74] . Cuando el espacio de estados o el espacio de acciones son discretos con un gran numero de variables o incluso continuos, las funciones Q deben de ser descritas de una forma aproximada ya que una representación exacta podría ser poco practica para fines de análisis , por lo proponemos un algoritmo de aprendizaje el cual usa una aproximación difusa de los estados junto con una versión modificada del algoritmo WoLF-PHC, para una tarea enteramente cooperativa, donde un vector ϕ es usado para parametrizar la función Q estado-acción.

4.2. Q iteración difusa para multi-agentes

El aproximador ϕ esta basado una partición difusa del espacio de estados, donde el espacio de acciones es asumido discreto. Existen N conjuntos difusos en la partición , los cuales están descritos por una función de membresía:

$$\mu_d(x) = X \to [0,1] \quad d = 1, 2, .., N$$

donde $\mu_d(x)$ describe el grado de membresía de el estado x hacia el conjunto difuso d, esta función de membresía puede ser vista como una función base [75]. Las funciones de membresías podrían tener algún tipo de significado lingüístico sí un conocimiento previo acerca de la función Q estuviera disponible, pero esto no es necesario en el presente método. El numero de funciones de membresía crece con la dimensionalidad del espacio de estados y del numero de agentes presentes en el entorno.

Las funciones de membresía triangulares son utilizadas en esta propuesta, ya que ellos presentan su máximo valor en un punto único, es decir, para todo conjunto difuso d existe

un único punto x_d (llamado el centro de la función de membresía) tal que :

$$\mu_d(x_d) > \mu_d(x) \forall x \neq x_d.$$

Ya que todas las demás funciones de membresías toman valores ceros en x_d, $\mu_{\hat{d}}(x_d) = 0$ para $\forall\ d \neq \hat{d}$ asumimos que$\mu_d(x_d) = 1$, lo que significa que las funciones de membresías son normales, la razón por escoger esta forma es que otras variantes de funciones de membresía que presenten valores muy grandes en x_d o que este valor no sea único, podrían generar divergencia al realizar la Q iteración [76].

Tendremos un numero de N funciones de membresías triangulares para cada estado x_e con $e = 1, 2, ..E$ con $\dim(X) = E$ para cada agente . Una forma piramidal de dimensión E sera la consecuencia de el producto de cada función de membresía unidimensional en la partición difusa del espacio de estados para cada agente presente :

$$\begin{aligned}
\mu_{e,1}(x_e) &= \text{máx}\left(0, \frac{c_{e,2} - x_e}{c_{e,2} - c_{e,1}}\right) \\
\mu_{e,d}(x_e) &= \text{máx}\left[0, \text{mín}\left(\frac{x_e - c_{e,d-1}}{c_{e,d} - c_{e,d-1}}, \frac{c_{e,d+1} - x_e}{c_{e,d+1} - c_{e,d}}\right)\right] \quad \text{para } d = 1, 2, .., N-1 \\
\mu_{e,N}(x_e) &= \text{máx}\left(0, \frac{x_e - c_{e,N-1}}{c_{e,N} - c_{e,N-1}}\right)
\end{aligned}$$

donde $c_{e,1}, c_{e,2}, ..., c_{e,N}$ son los núcleos a lo largo de la dimensión e y debe de satisfacer que $c_{e,1} < c_{e,2} < ... < c_{e,N.}$ El espacio de estados debe de estar contenido en el soporte de las funciones de membresía $x_e \in [c_{e,1}, c_{e,N}]$ para $e = 1, 2, .., E$. Funciones de membresías unidimensionales que sean adyacentes siempre se interseca en un valor de 0,5. Ejemplos de funciones de membresías unidimensional esta dado en la Figura 4.1 y de función de membresía bidimensional en la Figura 4.2.

Se asume que el espacio de acciones es discreto para todos los agentes en el entorno y que además cuentan con el mismo numero de acciones disponibles

$$U_i = \{u_{ij} \,|i = 1, 2, .., n \quad j = 1, 2, ..M\}$$

El aproximador difuso ϕ , el cual es un vector de parámetros con $z = nNM$ elementos a ser guardados, los pares funciones de membresía-acciones (μ_d, u_{ij}) para cada agente corresponden a elementos del vector de parámetros $\phi_{i,d,,j}$.

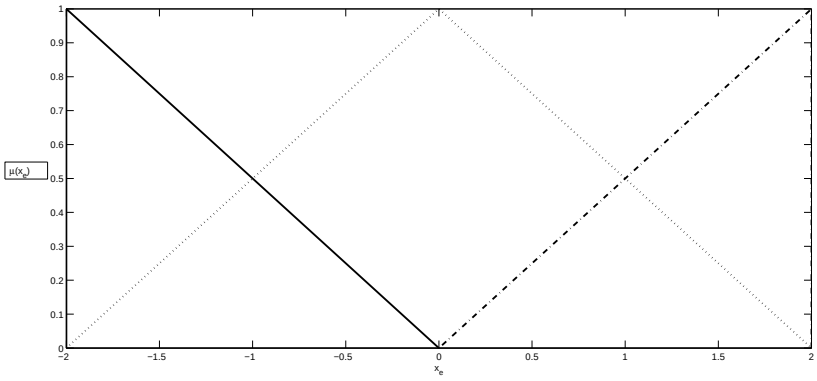

Figura 4.1: Conjunto de funciones de Membresia triangulares Unidimensional

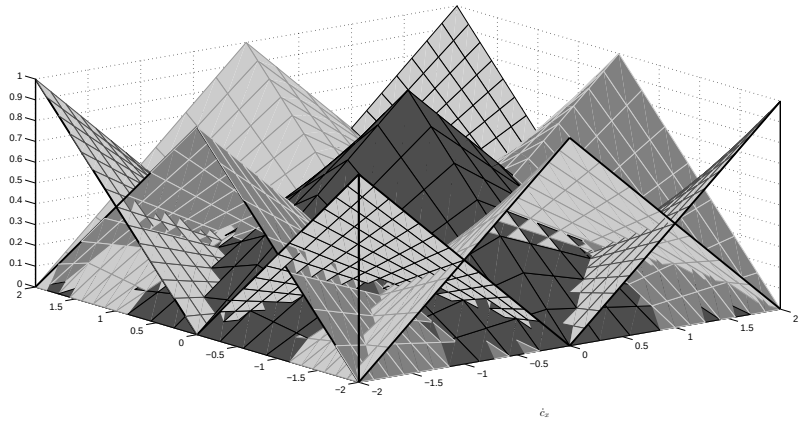

Figura 4.2: Funcion de membresias bidimensional al combinar dos conjuntos de funciones de membresias unidmiensionales.

Los elementos $\phi_{i,d,,j}$ del aproximador $\phi \in R^z$ son primero organizados preliminarmente en n matrices de tamaño $N \times M$, llenando las primeras M columnas con los N elementos disponibles. Posteriormente los elementos de las n matrices son colocados en un arreglo vectorial $\phi \in R^z$ columna a columna del primer agente, para luego proseguir con las columnas del segundo agente así hasta completar los n agentes.

De esta manera denotamos los índices escalares correspondientes a $[i, d, j]$ del vector de aproximación $\phi_{i,d,,j}$

$$[i, d, j] = [d + (j - 1) N] + (i - 1) N \times M$$

donde $i = 1, 2, .., n$ denota el numero del agente analizado , $d = 1, 2, .., N$ es el numero de particiones difusas del estado x_e con $e = 1, 2, .., E$ con $\dim(X) = E$, $j = 1, 2, .., M$ $\dim(U_i) = M$.

Los valores Q aproximados pueden ser calculados por medio de una suma ponderada de los parámetros $\phi_{i,d,,j}$

$$\tilde{Q}(x, \mathbf{u}) = \sum_{i=1}^{n} \sum_{d=1}^{N} \mu_d(x) \phi_{[i,d,j]} \tag{4.2}$$

El estado x es tomado como entrada por la base de reglas difusas produciendo M diferentes salidas para cada agente, los cuales son los correspondientes valores Q para cada acción disponible $u_{ij} | i = 1, 2, .., n \quad j = 1, 2, ..M$, por lo que las funciones de salida serán constantes y consistentes con los elementos $\phi_{i,d,,j}$. La bases de reglas difusas propuestas puede ser considerado como una base de reglas Takagi-Sugeno orden cero [77]:

$$\text{si } x \text{ es } \mu(x_d) \text{ entonces } q_{[i,1]} = \phi_{[i,d,1]}; ..; \; q_{[i,M]} = \phi_{[i,d,M]}$$

Una aproximación linealmente parametrizada de una función Q [78],emplea N funciones bases $\theta_1, \theta_2, .., \theta_N : X \times U \rightarrow R$ y un vector de parámetros ϕ z dimensional el cual es calculado con:

$$F(\phi) = (x, u) = \sum_{l=1}^{N} \theta_l(x, u) \phi_l = \theta^T(x, u) \phi \tag{4.3}$$

donde $\theta(x, u) = [\theta_1(x, u), \theta_2(x, u), ..., \theta_N(x, u)]$. La expresión (4.2) puede ser considerado como una aproximación linealmente parametrizada similar a la expresión (4.3) , este aproximador puede expresarse como un mapeo de aproximación $F = R^z \rightarrow \mathbf{Q}$, donde R^z es el

espacio de parámetros y $\mathbf{Q}$ es el espacio de las funciones Q , el parámetro ϕ provee de una representación de una función Q aproximada:

$$\tilde{Q}\left(x,\mathbf{u}\right)=\left[F\left(\phi\right)\right]\left(x,\mathbf{u}\right)=\sum_{i=1}^{n}\sum_{d=1}^{N}\mu_{d}\left(x\right)\phi_{[i,d,j]} \tag{4.4}$$

Por lo tanto en lugar de almacenar una gran cantidad de valores Q para los pares (x,u) de cada agente , lo cual es poco adecuado en espacio de estados continuos, solo almacenamos un numero de z elementos en el vector ϕ. El mapeo F solo representa un subconjunto de espacio de funciones $\mathbf{Q}$, ya que una función Q arbitraria cuando es aproximada lleva implícita un error de aproximación al usar el mapeo F [17].

Desde el punto de vista de la teoría aprendizaje por reforzamiento, las aproximaciones linealmente parametrizadas como el mapeo F son preferidas, ya que son mas adecuadas para analizar los aspectos teóricos del método propuesto [76]. Lo anterior fue la razón de nuestra elección para escoger la aproximación $\dot{\phi}$. Por lo que las funciones de membresías normalizadas μ_d pueden ser vistas como bases estado dependientes o características [79].

La expresión (4.4) provee función $\tilde{Q}$ el cual es una aproximación de la función Q exacta, por lo que la aproximación $\tilde{Q}$ es suministrada como una entrada al mapeo H, utilizando (4.4):

$$\bar{Q}_{l+1}\left(x,\mathbf{u}\right)=\left(H\circ F\right)\left(\phi_{l}\right)\left(x,\mathbf{u}\right)=H(\tilde{Q}\left(x,\mathbf{u}\right)) \tag{4.5}$$

En general a la nueva función $\bar{Q}\left(x,\mathbf{u}\right)$ no es posible guardarla en una forma explícita, como una alternativa $\bar{Q}\left(x,\mathbf{u}\right)$ tiene que ser representada en una forma aproximada [80], usando un nuevo vector de parámetros ϕ_{l+1} el cual es obtenido por medio de un mapeo de proyección $P:\mathbf{Q}\rightarrow R^{z}$:

$$\phi_{l+1}\left(x,\mathbf{u}\right)=P(\bar{Q}_{l+1})\left(x,\mathbf{u}\right) \tag{4.6}$$

el cual debe de asegurar que $\tilde{Q}_{l+1}\left(x,\mathbf{u}\right)=\left[F\left(\phi_{l+1}\right)\right]\left(x,\mathbf{u}\right)$ este tan cerca de $\bar{Q}_{l+1}\left(x,\mathbf{u}\right)$ como sea posible, en el sentido de una regresión de mínimos cuadrados :

$$P(Q)=\phi\ \text{ donde }\quad\phi\in\arg\min_{\phi}\sum_{\lambda}^{s}\left(Q\left(x_{\lambda},\mathbf{u}_{\lambda}\right)-F\left(\phi\right)\left(x_{\lambda},\mathbf{u}_{\lambda}\right)\right) \tag{4.7}$$

en el cual una serie de muestras estado-acción$(x, \mathbf{u})$ son usados. En la $Q-$iteración difusa propuesta, un conjunto de nNM muestras son utilizadas, el cual es obtenida como el producto entre el núcleo de las funciones de membresía y el conjunto de las acciones discretas. Debido a la utilización de funciones de membresías triangulares, la proyección (4.7) se convierte a un problema convexo de optimización cuadrática, por lo que se reduce a una asignación de la forma:

$$\phi_{[i,d,j]} = P(Q)_{[i,d,j]} = Q(x, \mathbf{u}) \tag{4.8}$$

Recapitulando, la Q-iteracion difusa aproximada comienza con un vector de parámetros $\phi \in R^z$ con valores arbitrarios, el cual es actualizado en cada iteración usando el mapeo

$$\phi_{l+1} = (P \circ H \circ F)(\phi_l) \tag{4.9}$$

y es detenido cuando el umbral ξ es mayor que la diferencia entre 2 sucesivas iteraciones del vector ϕ

$$\left\|\phi_{l+1} - \phi_l\right\| \leq \xi \tag{4.10}$$

Una política codiciosa (greedy) óptima puede ser obtenida desde ϕ^* (el cual es el vector de parámetros cuando $l \rightarrow \infty$), para cualquier estado, una acción es calculada por interpolación entre la mejor acción local para cada agente de todos los núcleos de las funciones de membresía x_d, mediante el uso de las funciones de membresías como pesos:

$$h_i^*(x) = \sum_{d=1}^{N} \phi_{i,d}(x)\, u_{j_{id}^*} \quad \text{donde } j_{i,d}^* \in \arg\max F(\phi^*)(x, \mathbf{u}) \tag{4.11}$$

La siguiente versión del mapeo H es usado en la implementación practica de Q-Iteración difusa:

$$[H(Q)](x, \mathbf{u}) = \rho(x, \mathbf{u}) + \gamma \max_j Q(f(x, \mathbf{u})\, \mathbf{u}_j)$$

donde cada iteración puede ser implementada como :

$$\phi_{l+1} = (P \circ H \circ F)(\phi_l) \tag{4.12}$$

Con la finalidad de implementar la actualización (4.9), en la siguiente sección se propone un procedimiento utilizando una versión modificada del algoritmo WoLF-PHC [3], de esta forma el algoritmo clásico puede ser extendido a problemas con sistemas multi-agentes con estados continuos pero con espacio de acción discretas, el algoritmo comienza con un valor arbitrario para $\phi(\phi = 0)$ hasta que un umbral de paro es alcanzado después de cierto numero de iteraciones.

Teorema 4.1 *La Q-iteracion difusa aproximada para sistemas multi- agentes presentado en el algoritmo 2 converge a un único punto fijo.*

Demostración. La convergencia de la Q-iteracion aproximada se garantiza al asegurar que el mapeo compuesto $P \circ H \circ F$ es una contracción en la norma infinita,en [72] se demuestra que el mapeo H es una contracción por lo que resta mostrar que los mapeos F y P son no expansiones.

Ya que el mapeo de aproximación F dado por $\left[F\left(\phi\right)\right]\left(x, \mathbf{u}\right) = \sum_{i=1}^{n}\sum_{d=1}^{N} \mu_d\left(x\right)\phi_{[i,d,j]}$ es una combinación lineal ponderada de las funciones de membresías tenemos:

$$\begin{aligned}
\left|F\left(\phi\right)\left(x, \mathbf{u}\right) - F\left(\phi^{'}\right)\left(x, \mathbf{u}\right)\right| &= \left|\sum_{i=1}^{n}\sum_{d=1}^{N} \mu_d\left(x\right)\phi_{[i,d,j]} - \sum_{i=1}^{n}\sum_{d=1}^{N} \mu_d\left(x\right)\phi^{'}_{[i,d,j]}\right| \\
= \left|\sum_{i=1}^{n}\sum_{d=1}^{N} \mu_d\left(x\right)\left[\phi_{[i,d,j]} - \phi^{'}_{[i,d,j]}\right]\right| &= \left|\sum_{i=1}^{n}\sum_{d=1}^{N} \mu_d\left(x\right)\right|\left|\phi_{[i,d,j]} - \phi^{'}_{[i,d,j]}\right| \\
\leq \sum_{i=1}^{n}\sum_{d=1}^{N} \left|\mu_d\left(x\right)\right|\left|\phi_{[i,d,j]} - \phi^{'}_{[i,d,j]}\right| &\leq \sum_{i=1}^{n}\sum_{d=1}^{N} \mu_d\left(x\right)\left|\phi_{[i,d,j]} - \phi^{'}_{[i,d,j]}\right| \\
\leq \sum_{i=1}^{n}\sum_{d=1}^{N} \mu_d\left(x\right)\left\|\phi - \phi^{'}\right\|_{\infty} &\leq \left\|\phi - \phi^{'}\right\|_{\infty}
\end{aligned}$$

Donde el ultimo paso es verdad ya que la suma de las funciones normalizadas es $\mu_d\left(x\right)$ es 1 y el producto generado por cada agente sera también 1, por lo que se demuestra que el mapeo F es una no expansión.

Ya que el mapeo P es una asignación $P(Q)_{[i,d,j]} = Q(x, \mathbf{u})$ y que las muestras son los centros de las funciones de membresías $\phi_l\left(x_l, \mathbf{u}_l\right) = 1$, $\phi_{l'}\left(x_l, \mathbf{u}_l\right)$ para toda $l^{'} \neq l$ el mapeo P es no expansivo a diferencia de una proyección por mínimos cuadrados que en general es una expansión [17].

El mapeo H es una contracción con $\gamma < 1$, por lo que $P \circ H \circ F$ es también una contracción con factor γ :

$$\begin{aligned} & \left\| (P \circ H \circ F)(\phi) - (P \circ H \circ F)\left(\phi^{'}\right) \right\| \\ \leq \; & \gamma \left\| (H \circ F)(\phi) - (H \circ F)\left(\phi^{'}\right) \right\|_{\infty} \\ \leq \; & \gamma \left\| F(\phi) - F\left(\phi^{'}\right) \right\|_{\infty} \\ \leq \; & \gamma \left\| \phi - \phi^{'} \right\|_{\infty} \text{ con } \gamma < 1 \end{aligned}$$

Por lo que $P \circ H \circ F$ tiene un punto fijo ϕ^* y el algoritmo mostrado converge a este punto fijo conforme $l \to \infty$. ■

Teorema 4.2 *Para cualquier elección del umbral $\xi > 0$ y cualquier valor inicial del vector de parámetros $\phi_0 \in R^z$, la Q-iteracion difusa propuesta en el algoritmo 2 es terminada en un numero finito de iteraciones*

Demostración. Como se demostró en el teorema 1, el mapeo $P \circ H \circ F$ es una contracción con factor $\gamma < 1$ con un punto fijo ϕ^*

$$\left\| \phi_{l+1} - \phi^* \right\|_{\infty} = \left\| (P \circ H \circ F)(\phi_l) - (P \circ H \circ F)(\phi^*) \right\| < \gamma \left\| \phi_l - \phi^* \right\|_{\infty}$$

Entonces si $\left\| \phi_{l+1} - \phi^* \right\|_{\infty} < \gamma \left\| \phi_l - \phi^* \right\|_{\infty}$ por inducción tenemos $\left\| \phi_l - \phi^* \right\|_{\infty} < \gamma^l \left\| \phi_0 - \phi^* \right\|_{\infty}$ para $l > 0$. De acuerdo al teorema de Banach de punto fijo tenemos que ϕ^* es acotada. Ya que el vector con el que se inicia la iteración es acotado , entonces $\left\| \phi_0 - \phi^* \right\|_{\infty}$ es también acotado.

Sea $G_o = \left\| \phi_0 - \phi^* \right\|_{\infty}$ el cual es acotado y que $\left\| \phi_l - \phi^* \right\|_{\infty} \leq \gamma^l G_0$ para $l > 0$, aplicando la desigualdad del triángulo :

$$\begin{aligned} \left\| \phi_{l+1} - \phi_l \right\|_{\infty} & \leq \left\| \phi_{l+1} - \phi^* \right\|_{\infty} + \left\| \phi_l - \phi^* \right\|_{\infty} \\ & \leq \gamma^{l+1} G_0 + \gamma^l G_0 = \gamma^l G_0 \left[\gamma + 1 \right] \end{aligned}$$

Si $\gamma^l G_0 [\gamma + 1] = \xi$

$$\gamma^l = \frac{\xi}{G_0 [\gamma + 1]}$$

Aplicando logaritmo base γ a ambos lado de la expresión anterior:

$$l = \log_\gamma \left[\frac{\xi}{G_0 [\gamma + 1]} \right]$$

con $G_o = \|\phi_0 - \phi^*\|_\infty$ la cual es acotada y $\gamma < 1$ implica que l es finita. Por lo que el algoritmo 2 es detenido en lo mas l iteraciones. ■

4.3. Planeación de Trayectoria

Se propone un algoritmo con la intención de implementar el aprendizaje por reforzamiento a sistemas multi- agentes utilizando una iteración difusa aproximada dada la expresión (4.9), en este la función de transición de estados f y la función de recompensa ρ son consideradas conocidas:

Algoritmo 2:

1. Sea $\alpha \in (0, 1]$, el factor de aprendizaje $\delta_l > \delta_w \in (0, 1]$, se inicializa :

$$\phi(x, \mathbf{u}) \longleftarrow 0, \pi(x, u) \longleftarrow \frac{1}{|U_i|} \tag{4.13}$$

   donde $\pi(x, u)$ es la probabilidad de escoger la acción u en el estado x, $|U_i|$ es la cardinalidad del conjunto U .

2. Se repite

   - Para el estado x, seleccionamos la acción u de acuerdo a una estrategia $\pi(x)$ con una exploración adecuada. En cada paso de iteración un acción aletearía con probabilidad $\varepsilon \in (0, 1)$ es utilizada.

- Aplicar la Q-iteracion difusa, para funciones de membresías $\mu_d \;\; d = 1, .., N$ y acciones discretas $U_j \quad j = 1, ..., M$, seleccionamos el umbral $\xi > 0$

$$\phi_{[i,d,j]} \longleftarrow \rho\left(x, \mathbf{u}\right) + \gamma \max_{j'} \sum_{i=1}^{n} \sum_{d'=1}^{N} \mu_{d'}\left(f\left(x, \mathbf{u}\right)\right) \phi_{\left[i,d',j'\right]} \tag{4.14}$$

- Hasta cumplir :

$$\left\|\phi_{l+1} - \phi_l\right\| \leq \xi$$

3) - Actualizar el promedio $\bar{\pi}$

$$\begin{aligned} &C(x) \longleftarrow C(x) + 1 \\ &\bar{\pi}(x, u^{'}) \longleftarrow \tfrac{1}{C(x)}\left(\pi(x, u^{'}) - \bar{\pi}(x, u^{'})\right) \end{aligned} \tag{4.15}$$

donde $\forall u^{'} \in U_i$,

- Actualizar la estrategia π

$$\pi(x, u) \longleftarrow \pi(x, u) + \Delta_{xu} \tag{4.16}$$

$$\begin{aligned} &\Delta_{xu} = -\delta_{xu} && \text{si } u \neq \arg\max_{u'} \phi(x, u^{'}) \\ &\Delta_{xu} = \textstyle\sum_{u' \neq u} \delta_{xu'} && \text{de otro modo} \end{aligned} \tag{4.17}$$

con $\delta_{xu} = \min\left(\pi(x, u), \frac{\delta}{|U_i| - 1}\right)$

$$\begin{aligned} &\delta = \delta_w \quad \text{si } \textstyle\sum_{u'} \pi(x, u^{'})\phi(x, u^{'}) > \bar{\pi}(x, u^{'})\phi(x, u^{'}) \\ &\delta = \delta_l \qquad\qquad\qquad \text{de otro modo} \end{aligned} \tag{4.18}$$

- Salida :

$$\phi^{*} = \phi_{l+1}$$

Una política codiciosa (greedy) es obtenida por medio

$$h_i^*(x) = \sum_{d=1}^{N} \phi_{i,d}(x)\, u_{j_{id}^*} \quad \text{donde } j_{i,d}^* \in \arg\max F(\phi^*)(x, \mathbf{u}) \tag{4.19}$$

Donde $j_{i,d}^*$ corresponde a una acción localmente óptima para el núcleo x_d de la función de membresía correspondiente para el agente i. La expresión dado (4.14) corresponde a $H(\tilde{Q}(x, \mathbf{u}))$ donde $\tilde{Q}(x, \mathbf{u}) = F(\phi)$, de acuerdo con la expresión (4.8) donde $\phi_{[i,d,j]} = P(Q)_{[i,d,j]} = Q(x, \mathbf{u})$, de esta forma la Q-iteracion difusa $\phi_{l+1} = (P \circ H \circ F)(\phi_l)$ es realizada.

4.4. Resultados en simulación

Con la finalidad de validar el algoritmo propuesto en la sección anterior, se realiza una simulación utilizando el software Matlab [62]. La Q-iteracion difusa aproximada se aplica a una tarea cooperativa donde están presentes 2 agentes en un entorno bidimensional con variables de estados continuas y acciones discretas. Los dos agentes con masa m tienen que ser conducidos sobre una superficie plana de tal manera que ambos lleguen al mismo tiempo al punto origen u objetivo lo mas rápido posible.

En esta tarea el vector de estados $x = [x_1, x_2, ..., x_8]^T$ contiene las coordenadas bidimensionales de cada agente s_{ix} , s_{iy} y sus respectivas velocidades $\dot{s}_{ix}$, $\dot{s}_{iy}$ para $i = 1, 2$, donde i denota el agente

$$x = \left[s_{1x}, s_{1y}, \dot{s}_{1x}, \dot{s}_{1y}, s_{2x}, s_{2y}, \dot{s}_{2x}, \dot{s}_{2y}\right]^T$$

El movimiento de los agentes puede ser afectado por la influencia de la fricción, haciendo las

dinámicas del sistema no lineales. La dinámica continua del sistema es dado por

$$\begin{aligned} \ddot{s}_{1x} &= -\eta\left(s_{1x}, s_{1y}\right)\frac{\dot{s}_{1x}}{m_1} + \frac{u_{1x}}{m_1} \\ \ddot{s}_{1y} &= -\eta\left(s_{1x}, s_{1y}\right)\frac{\dot{s}_{1y}}{m_1} + \frac{u_{1y}}{m_1} \\ \ddot{s}_{2x} &= -\eta\left(s_{2x}, s_{2y}\right)\frac{\dot{s}_{2x}}{m_2} + \frac{u_{2x}}{m_2} \\ \ddot{s}_{2y} &= -\eta\left(s_{2x}, s_{2y}\right)\frac{\dot{s}_{2y}}{m_2} + \frac{u_{2y}}{m_2} \end{aligned} \tag{4.20}$$

Donde $\eta\left(s_{ix}, s_{iy}\right)$ para $i = 1, 2$ es la función de fricción el cual depende la posición de cada agente, la señal de control $\mathbf{U} = \left[u_{1x}, u_{1y}, u_{2x}, u_{2y}\right]^T$ es una fuerza, y m_i para $i = 1, 2$ es la masa de cada agente.

Con el objetivo de obtener la función de transición de estados f, se realiza un discretización con un paso de $T = 0{,}4$ segundos y la expresión que describe las dinámicas del sistema son integradas entre el tiempo de muestreo. En la simulación seleccionamos el punto inicial del recorrido de manera aletearía y llevamos acabo 1000 iteraciones de aprendizaje utilizando el algoritmo 2, en el caso de alcanzar el limite de las 1000 iteraciones sin poder completar el objetivo el experimento es reiniciado.

Las magnitudes de las variables de estado y acción son acotadas de la siguiente manera; s_{ix} y $s_{iy} \in [-6, 6]$ metros, $\dot{s}_{ix}$ y $\dot{s}_{iy} \in [-3, 3]\,\frac{m}{s}$, también la señal de control es acotada $u_{ix}, u_{ix} \in [-2, 2]$ para $i = 1, 2$. El coeficiente de fricción es tomado constante $\eta = 1\frac{kg}{s}$, las masas de los agentes es tomada iguales para ambos en $m = 0{,}5$ kg.

Las acciones de control para cada agente son discretizadas con 25 elementos $U_i = [-2\,{-}0{,}2\ 0\ 0{,}2\ 2] \times [-2\,{-}0{,}2\ 0\ 0{,}2\ 2]$ para $i = 1, 2$, los cuales corresponden a la fuerza aplicada en forma diagonal, hacia la izquierda, derecha, hacia arriba, hacia abajo y ninguna fuerza aplicada.

Las funciones de membresías utilizadas para las variables de estado correspondientes a la posición y velocidad tienen una forma triangular , donde el centro de las funciones de membresías x_d determinan las formas de ellas. Los centros de las funciones de membresía para las variables de posición s están localizadas en $[-6, -3, -0{,}3, -0{,}1, 0, 0{,}1, 0{,}3, 3, 6]$ y los centros para las variables de estado de velocidad $\dot{s}$ están localizadas en $[-3, -1, 0, 1, 3]$ lo anterior es valido para ambos agentes.

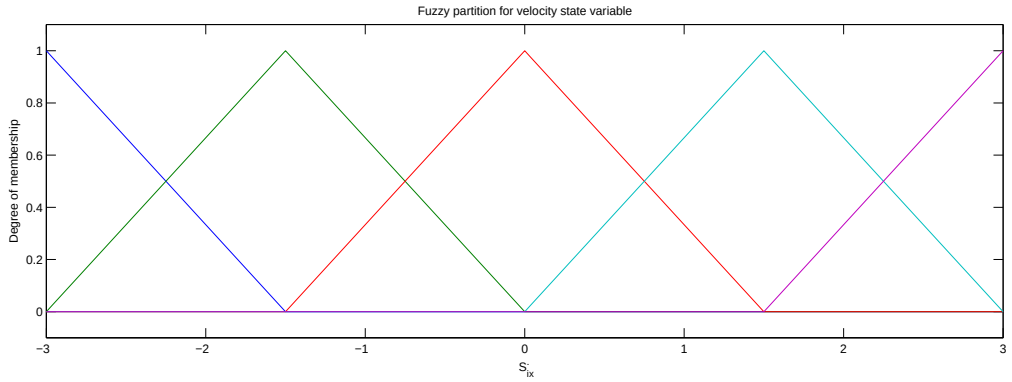

Figura 4.3: Particion trangular difusa usada en la variable de estado de velocidad $\dot{s}$

Por lo anterior 50625 pares estado-acción $(x, \mathbf{u})$ son aproximados por cada agente en el vector de parámetros ϕ, la cantidad de pares estado-acción aumenta en función del numero de funciones de membresía presentes, es de notar que a un mayor numero de funciones de membresía se genera una mayor resolución en las acciones de control. Un ejemplo de las particiones triangulares difusas empleadas se muestra en la Figura 4.3.

Las particiones del espacio de estado x se determina por el producto de las funciones de membresías individuales de cada agente :

$$\mu(x) = \prod_{i=1}^{2} \mu_{s_{ix}} \prod_{i=1}^{2} \mu_{s_{iy}} \prod_{i=1}^{2} \mu_{\dot{s}_{ix}} \prod_{i=1}^{2} \mu_{\dot{s}_{iy}} \tag{4.21}$$

El objetivo final de arribar al mismo tiempo al punto origen en un mínimo tiempo transcurrido es representado por la función de recompensa ρ común a ambos agentes:

$$\begin{aligned} \rho\,(x, \mathbf{u}) &= 5 \text{ si } \|x\| < 0{,}1 \\ \rho\,(x, \mathbf{u}) &= 0 \text{ in otro modo} \end{aligned} \tag{4.22}$$

En relación con el problema de coordinación, en el algoritmo propuesto los agentes toman en cuenta uno al otro en una forma de coordinación implícita, donde estos aprenden a preferir soluciones igualmente buenas y después descartar las demás opciones.

Después que la Q-iteracion difusa se realiza, el vector de parámetros óptimos ϕ^* se obtiene, por lo que una política de acciones puede ser derivada a través de una interpolación entre las mejores acciones locales disponibles para cada agente

$$h_i^*(x) = \sum_{d=1}^{N} \phi_{i,d}(x)\, u_{j_{id}^*} \quad \text{donde } j_{i,d}^* \in \arg\max F(\phi^*)(x, \mathbf{u}) \tag{4.23}$$

Para esta simulación, los parámetros utilizados en el algoritmo 2 fueron: el factor de descuento $\gamma = 0{,}96$, el umbral de paro $\xi = 0{,}05$ y las condiciones iniciales del sistema para la simulación fueron $s_0 = [-4, -6, -2, 2, 5, 3, 2, -1]$, la convergencia se presenta después de 310 iteraciones.

La Figura 4.4 muestra el desarrollo de los estados y la acción de control empleada por el agente 1, mientras que la Figura 4.5 muestra el desarrollo de los estados y la acción de control empleada por el agente 2. La trayectoria final elegida por los agentes es mostrada en la Figura 4.6.

La selección de los valores de los parámetros γ y ξ fueron seleccionadas de manera arbitraria, el vector de parámetros ϕ converge después de 330 iteraciones, donde la cota $\left\|\phi_{l+1} - \phi_l\right\| \leq \xi$ es alcanzada.

En la trayectoria final mostrada en la Figura 4.6, se muestra evidentemente una diferencia entre una trayectoria óptima, la cual conduciría a ambos agentes en una línea recta hasta el origen al partir desde cualquier posición inicial, lo anterior se debe a la elección de la cuantización difusa utilizada, ya que ante un mayor numero de funciones de membresías presente mejor sera la resolución del controlador, pero aumentara el numero de elementos a ser aprendidos, por lo que la trayectoria final obtenida es la mejor que puede ser conseguida con la presente partición difusa.

4.5. Resultados Experimentales

Para la implementación experimental del método propuesto, se utilizaron 2 robots móviles Khepera III, los cuales se componen por 2 ruedas pequeñas y 5 pares sensores ultrasonicos,

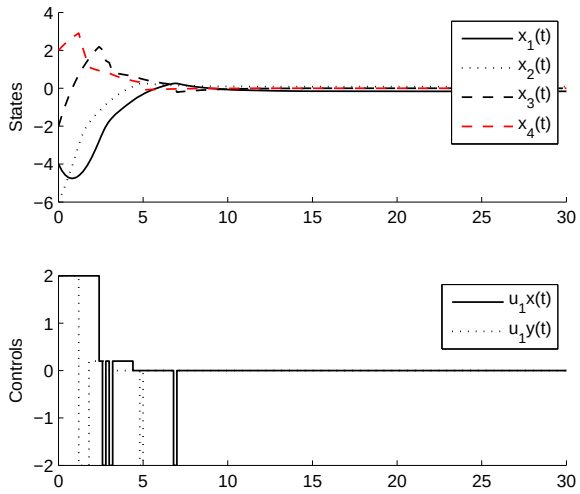

Figura 4.4: Estados y señal de control para el agente 1

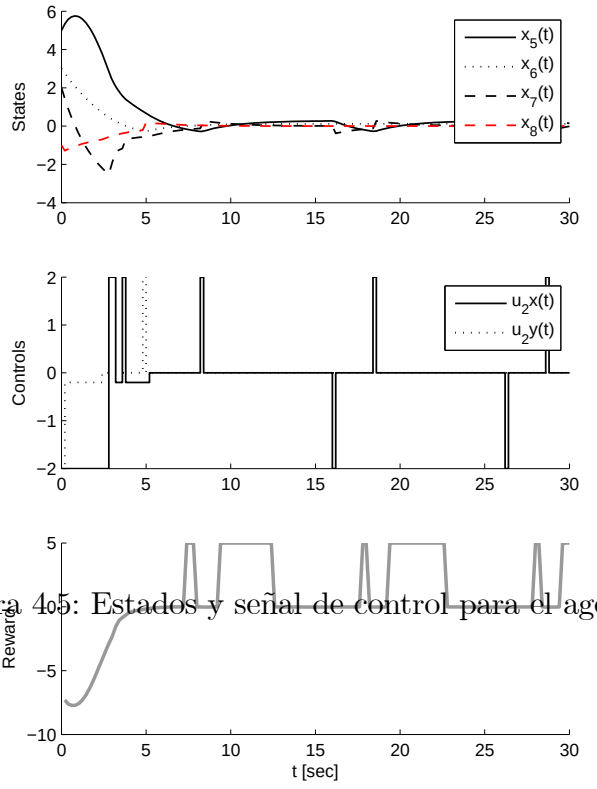

Figura 4.5: Estados y señal de control para el agente 2

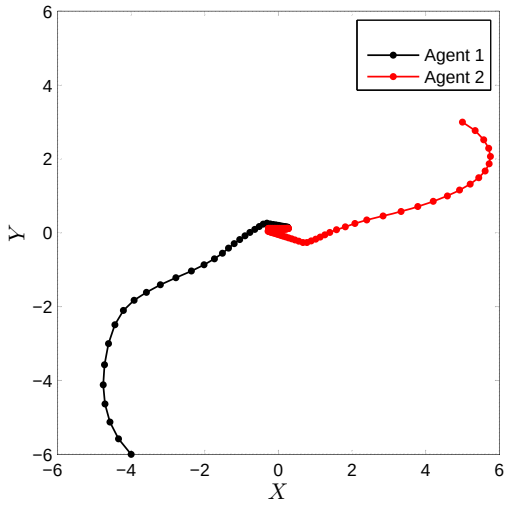

Figura 4.6: Trayectoria final por el Agente 1 y Agente 2

donde cada par esta integrado por un emisor y un receptor de señal., los cuales son utilizados para detectar las características físicas del entorno tales como obstáculos y otros agentes, la Figura 4.7 muestra el dispositivo utilizado.

Las lecturas de los sensores ultrasonicos $l_{a,c}$ son cuantificados en 3 particiones, los cuales representa la distancia entre el agente y las demás entidades presentes en el entorno, donde 0 indica que los demás agentes u obstáculos se encuentran cerca, el numero 1 indica que están a una distancia media y el numero 2 que se encuentran relativamente lejos de los sensores. En adición, se agregan 2 parámetros, el termino d_a el cual cuantifica la distancia que existe entre en agente y la posición de goal final y el termino p_a el cual determina el ángulo relativo entre el agente y el objetivo del goal final (medido en contra de las manecillas del reloj), ambos parámetros son divididos en 9 particiones que van del 0 al 8. Donde 0 representa la distancia o ángulo mas pequeño y en numero 8 indica la mas grande distancia o ángulo entre el agente y la posición final objetivo.

Las acciones disponibles para los Robots Móviles Khepera III son:

Figura 4.7: Robot Movil Khepera III

- Moverse hacia adelante
- Dar giro en sentido del reloj
- Dar giro en contra del sentido del reloj
- Mantenerse quieto

Durante el proceso de experimentación las lecturas de los sensores fueron en general fluctuantes, lo que nos daría como resultado una descripción imprecisa de los estados de los agentes y por lo tanto generar una convergencia errónea al realizar la actualización de los valores Q mediante la iteración difusa. Por lo que el efecto anterior se minimizo por medio de permitir un periodo de espera después de que los agentes realicen una acción conjunta coordinada y de esta manera asegurarnos que los sensores de los robots móviles tengan lecturas estables antes de que ellas sean utilizadas por el algoritmo de aprendizaje.

En adición, también se procuró que los robots Khepera se movieran a una velocidad relativamente baja durante el proceso de aprendizaje, con la intención de reducir el numero colisiones con objetos u otros agentes presentes, es importante señalar que se buscó que la cuantificación seleccionada de las lecturas de los sensores fuera suficiente para representar las posiciones y velocidades de manera adecuada [61].

Para validar el algoritmo propuesto, se realizó un experimento donde el objetivo de la tarea cooperativa fue que los 2 robots móviles Khepera III llegarán a una posición origen en el mismo momento, buscando que transcurriera el menor tiempo posible desde su salida. La iteración difusa aproximada se aplicó a un entorno con estados continuos y variable de acción discretas. La partición difusa utilizada para describir los estados del sistema fueron similares que las usadas en las simulaciones de la sección precedente, por lo que el proceso de aprendizaje estará sujeto a las lecturas de los sensores los cuales pueden ser perturbados por fuentes externas tales como luces, sombras, fuentes de calor etc.

El vector de estados $x = [x_1, x_2, ..., x_8]^T$ contiene las coordenadas bidimensionales de cada agente s_{ix} y s_{iy} y sus velocidades bidimensionales $\dot{s}_{ix}$ y $\dot{s}_{iy}$ para $i = 1, 2$:

$$x = [s_{1x}, s_{1y}, \dot{s}_{1x}, \dot{s}_{1y}, s_{2x}, s_{2y}, \dot{s}_{2x}, \dot{s}_{2y}]^T$$

La señal de control es $\mathbf{U} = [u_{1x}, u_{1y}, u_{2x}, u_{2y}]^T$ donde $\mathbf{U}$ es la fuerza aplicada a cada robot. Esta fuerza es convertida a una señal de pulsos con la intención de accionar los motores de CD de los robots, las magnitudes del espacio de estados y del espacio de acciones se consideran acotadas , s_{ix} y $s_{iy} \in [-6, 6]$ metros, $\dot{s}_{ix}$ y $\dot{s}_{iy} \in [-1, 1]\,\frac{m}{s}$, también la fuerza es acotada $u_{ix}, u_{ix} \in [-0{,}4, 0{,}4]$ para $i = 1, 2$.

La señal de control es discretizada con 25 elementos $U_i = [-0{,}4 -0{,}2\ 0\ 0{,}2\ 0{,}4] \times [-0{,}4 -0{,}2\ 0\ 0{,}2\ 0{,}4]$ para $i = 1, 2$. Fueron usadas funciones de membresias con forma triangular donde cada centro esta denotado por x_d, los centros de las funciones de membresía para el dominio de la posición s fueron centrados en $[-6, -3, -0{,}3, -0{,}1, 0, 0{,}1, 0{,}3, 3, 6]$ y los centros para el dominio de la velocidad $\dot{s}$ se localizan en $[-1, -0{,}5, 0, 5, 1]$ para cada agente.

El objetivo de la tarea cooperativa se describe mediante la función de recompensas común ρ:

$$\begin{aligned} \rho(x, \mathbf{u}) &= 5 \text{ si } \|x\| < 0{,}2 \\ \rho(x, \mathbf{u}) &= 0 \text{ de otro modo} \end{aligned} \tag{4.24}$$

Para este experimento el factor de descuentos se estableció en $\gamma = 0{,}96$ y el umbral de paro de algoritmo en $\xi = 0{,}2$. Las condiciones iniciales de los robots se muestran en la Figura 4.8, las cuales son:

$$x_0 = [-4, 5, 0, 0, 5, 3, 0, 0]$$

La convergencia del algoritmo propuesto hacia los valores óptimos estado-acción se da después de 275 iteraciones, en la Figura 4.9 se muestra los estados, la señal de control $U_1 = [u_{1x}, u_{1y}]$ y la recompensa obtenida para el agente 1, mientras que la Figura 4.10 muestra los estados, la señal de control $U_2 = [u_{2x}, u_{2y}]$ y la recompensa obtenida para el agente 2.

La trayectoria final seguida por ambos robots móviles es mostrada en la Figura 4.11, la cual muestra claramente que no es una línea recta como se podría suponer para una trayectoria óptima, lo anterior es debido al tipo de cuantificación difusa utilizado para describir los estados del sistema y al efecto de la fricción entre el piso y las ruedas del robot, sin embargo, con la cuantificación usada en este experimento la trayectoria final obtenida es la mejor que puede ser lograda. Los resultados obtenidos podrían mejorar mediante un mayor numero de funciones de membresias presentes para describir los estados.

Figura 4.8: Posiciones iniciales del experimento

4.6. Comparación de resultados con algoritmo CMOMMT

Existen otros métodos de aprendizaje por reforzamiento en sistemas multi-agentes que son capaces de manejar espacio de estados continuos, pero que se encuentran limitados a un determinado tipo de tareas, uno de estos métodos es el algoritmo llamado Çooperative multi-robot observation of multiple moving target"(CMOMMT) descrito en [67], el cual se basa en información local disponible para los agentes, con la intención de aprender conductas cooperativas en tareas con espacios de estados continuos.

El tipo de aprendizaje cooperativo utilizado en el algoritmo CMOMMT es mediante técnicas implícitas, lo que permite reducir la descripción del espacio de estados, a través de un mapeo del estado continuo a una representación finita de los estados, donde cada nuevo estado discretizado es considerado como una región del espacio continuo original. El espacio de acciones es discretizado en conjuntos adecuados para cada tipo de tarea encomendada,

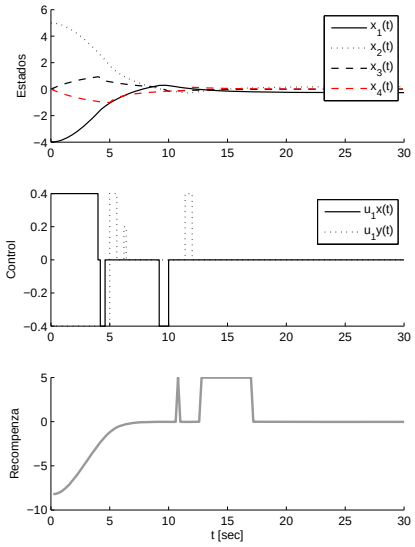

Figura 4.9: Estado, senal de control y recompenza obtenido por el Agente 1

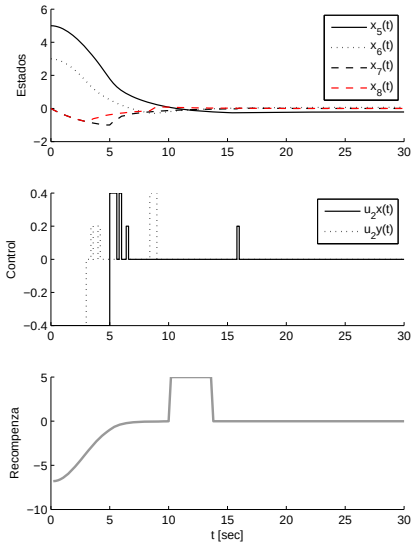

Figura 4.10: Estado, senal de control y recompenza obtenido por el Agente 2

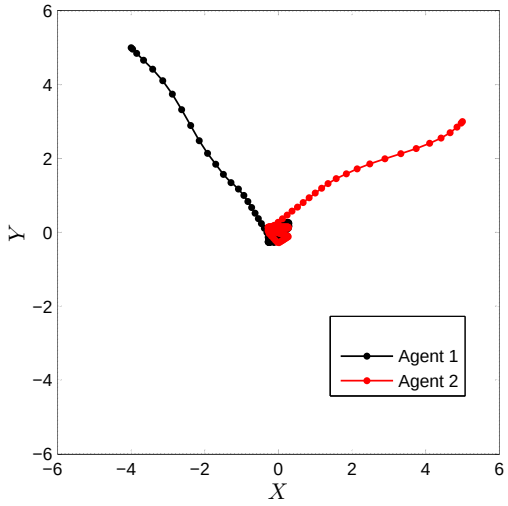

Figura 4.11: Trayectoria experimental seguida por el agente 1 y el agente 2

además, utiliza una señal de recompensa con retraso donde valores positivos o negativos son obtenidos al final del entrenamiento. Finalmente se obtiene una política de acciones óptimas mediante el uso de técnicas de clustering en espacio de acciones discreto.

Se realizó la misma tarea cooperativa de arribar los 2 agentes al mismo tiempo al punto origen en el menor tiempo transcurrido como en la sección previa, con la idea de que sea el experimento comparable, se usó la misma función de recompensa y la misma representación del espacio de estados continuo. Las condiciones iniciales fueron establecidas :

$$x_0 = [-4, 5, 0, 0, 5, -3, 0, 0]$$

La trayectoria final obtenida por el algoritmo CMOMMT se muestra en la Figura 4.12, donde se observa que esta no es una trayectoria recta y cerca del punto de meta muestra una oscilación persistente en las posiciones de los agentes. Los estados de los agentes se muestran en las Figuras 4.13 y 4.14 . El algoritmo mostró convergencia después de 310 iteraciones. Una posible razón de los poco favorables resultados obtenidos, puede ser que las funciones Q obtenidas son menos suaves que las presentadas por la aproximación difusa de la sección anterior, esto debido al proceso de discretización llevado acabo en el espacio de estados por el algoritmo CMOMMT, de esta manera el método de iteración difusa propuesto en este trabajo muestra un mejor desempeño en la forma de una representación mas exacta del espacio de estado y de un menor uso de poder computacional.

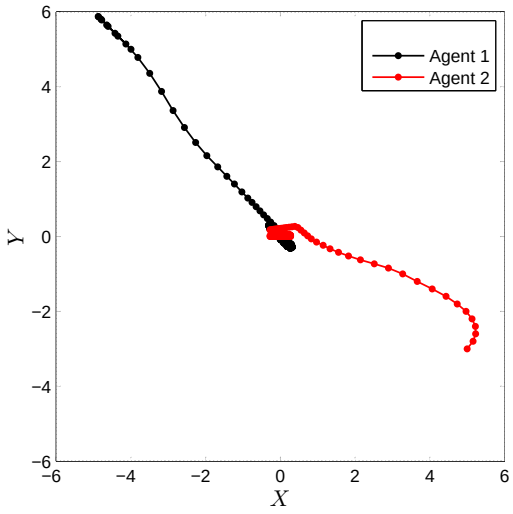

Figura 4.12: Trayectoria obtenida por algoritmo CMOMMT

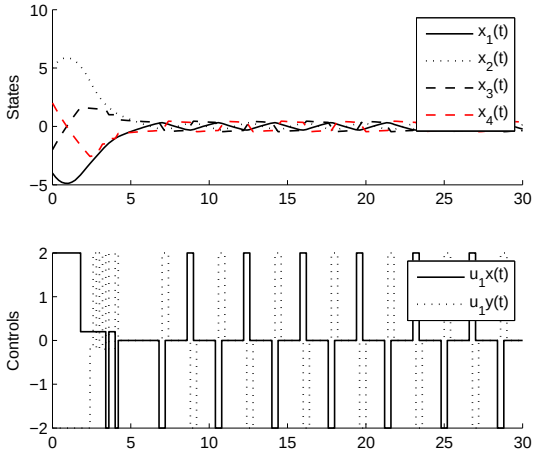

Figura 4.13: Estados y señal de control agente 1, algoritimo CMOMMT

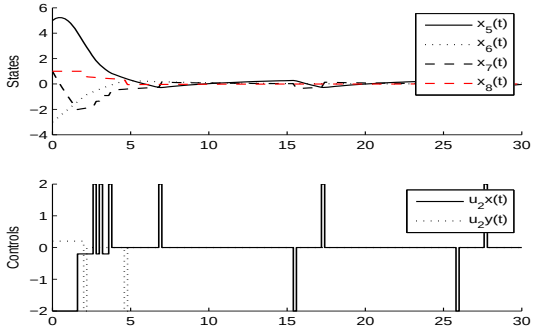

Figura 4.14: Estados y señal de control del agente 2, algoritmo CMOMMT

Capítulo 5

Conclusiones y trabajo a futuro

Uno de las principales direcciones de investigación en el área de inteligencia artificial es el desarrollo de robots móviles autónomos con la capacidad de desarrollar habilidades cooperativas, la gran mayoría de las aplicaciones reales están descritos mediante espacios continuos y sujetos a cambios dados por la interacción de los agentes con el entorno, por lo anterior en la presente tesis se proponen dos enfoques los cuales tienen la intención de mejorar el desempeño en el aprendizaje por reforzamiento.

En el presente trabajo se ha presentado distintas propuestas para lidiar con el problema de dimensionalidad, propio de los algoritmos de aprendizaje por reforzamiento en sistemas multi-agentes, la primera propuesta esta basada en el algoritmo clásico WoLF-PHC [3] en donde le fue añadido una segunda etapa de aprendizaje la cual utiliza las características de aproximación no lineal de las redes neuronales artificiales y de los Kernel Smoothers, lo anterior permite obtener una política de acciones al estar los agentes en estados que no habían sido visitados durante la primera fase de aprendizaje, evitando volver a ejecutar el algoritmo de aprendizaje, ahorrando tiempo y esfuerzo computacional

Mediante simulaciones en computadora y experimentación se comprobó la viabilidad de la presente propuesta en una tarea cooperativa para sistema multi agente, donde el estado

inicial de los agentes es desconocido (no había sido visitado previamente y no proveer datos la Q-tabla acerca de las acciones óptimas), por medio de la red neuronal entrenada y del kernel se obtuvieron una política de acciones sub-óptimas que permitió a los agentes completar la tarea asignada. Como trabajo a futuro estaría realizar una extensión de la anterior propuesta a sistemas con estados y acciones continuas, los cuales se asemejan a los problemas reales encontrados en la vida diaria.

La segunda propuesta presentada,es una aproximación de la Q-función, por medio de un vector de parámetros , para llegar a cabo lo anterior se recurrió a una partición difusa de los estados del sistema, lo que permite manejar estados continuos,sin la necesidad de guardar los valores estado-acción en tablas de busqueda, se mantuvo la premisa de que el conjunto de acciones disponibles para los agentes es discreta. Las funciones de membresias tuvieron forma triangular con la intención de facilitar el análisis matemático de convergencia del algoritmo,además, se presentaron 2 teoremas donde garantizan la convergencia del algoritmo propuesto a un punto fijo en un numero acotado de iteraciones.

La viabilidad del algoritmo anterior fue comprobado por medio de simulación en computadora y en una tarea experimental utilizando 2 robots Khepera III desempeñando una tarea cooperativa, donde estos 2 agentes tienen que llegar a punto meta al mismo tiempo en el menor tiempo transcurrido, se menciona que para tener una mejor resolución de partición de los estados y obtener una mejor política de acciones es necesario un mayor numero de funciones de membresía, los cuales incrementan el numero de elementos en el vector de aproximacion. Se realizó una comparativa con el algoritmo CMOMMT el cual es capaz de manejar entornos con estados continuos, la cual mostró que nuestra propuesta presenta un mejor desempeño bajo condiciones especificas, con la idea de que ambos resultados fuesen comparables.

En la parte experimental, se utilizaron 2 robots Khepera III para llevar a cabo las tareas cooperativas, los algoritmos de aprendizaje se realizaron en una computadora portátil de manera off-line y la comunicación con los robots fue a través de Bluetooth. Los sensores de obstáculos de dichos robots se mostraron fácilmente perturbables en la presencia de factores externos, la fricción entre el piso y la base de los robots genero problemas durante los

experimentos llevándonos a falsos puntos de convergencia.

Entre los posibles trabajos a futuro estaría generar una extensión a la propuesta anterior en la forma de incluir sistemas con un conjunto de acciones continuas, sistemas con dinámicas estocásticas o mas aún sistemas en donde la dinámica no fuera conocida, ya que el presente método se basa en un conocimiento previo de la tarea y del sistema en forma de la función de transición estados f y de la función de recompensa ρ , todo ello con la finalidad de volver al algoritmo un proceso en línea.

Bibliografía

[1] P. Stone and M. Veloso, "Multiagent systems: A survey from machine learning perspective ", *Autonomus Robots,* Vol.8, No.3, 345-383, 2000

[2] J. Park, J-H. Kim, J-B. Song, "Path Planning for a Robot Manipulator based on Probabilistic Roadmap and Reinforcement Learning", *International Journal of Control, Automation, and Systems*, Vol.5, No.6, 674-680, 2007

[3] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate", *Artificial Intelligence*, Vol.136, No.2, 215-250, 2002.

[4] L. P. Kaelbling , M. L. Littman and A. W. Moore, Reinforcement Learning: A Survey", *Journal of Artificial Intelligence Research*, Vol.4, No.1, 237-285, 1996

[5] C. Watkins and P. Dayan, "Q Learning: Technical Note ", *Machine Learning ,* Vol.8, 279-292, 1992

[6] R. Sutton and A. Barto, *Reinforcement Learning: an introduction*. Cambridge, MA : MIT Press, 1998

[7] Boutilier C. "Planning, Learning and Coordination in Multiagent Decision Processes", *In Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge (TARK96). 1996 . pages 195-2102*

[8] M.L. Littman "Value-function reinforcement learning in Markov games". *Journal of Cognitive Systems Research* 2(1), 55–66 (2001)

[9] A.G Barto, R.S Sutton and C.W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems". *IEEE Transactions on System, Man and Cybernetics* 13(5) 833-846 (1983)

[10] R.S. Sutton "Learning to predict by the method of temporal differences ". *Machine Learning* 3, 9–44 (1988)

[11] J. Peng, R.J Williams , "Incremental multi-step Q-learning ". *Machine Learning* 22(1–3), 283–290 (1996)

[12] M.L Puterman "Markov Decision Processes—Discrete Stochastic Dynamic Programming ". Wiley, Chichester (1994)

[13] A.W. Moore and C.G. Atkeson "Prioritized sweeping: Reinforcement learning with less data and less time ". *Machine Learning* 13, 103–130 (1993)

[14] R.S. Sutton "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming "*Proceedings 7th International Conference on Machine Learning* (ICML-1990), Austin, US, pp. 216–224 (1990)

[15] T. Jaakkola , M.I. Jordan and S.P Singh .On the convergence of stochastic iterative dynamic programming algorithms". *Neural Computation* 6(6), 1185–1201 (1994)

[16] Y. Ishiwaka , T. Sato and Y. Kakazu , .An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning". *Robotics and Autonomous Systems* 43(4), 245–256 (2003)

[17] J.N. Tsitsiklis , .Asynchronous stochastic approximation and Q-learning ". *Machine Learning* 16(1), 185–202 (1994)

[18] M.P. Wellman , A.R. Greenwald , P. Stone and P.R. Wurman ,"The 2001 Trading Agent Competition ". *Electronic Markets* 13(1) (2003)

[19] T. Jung , D. Polani , "Kernelizing LSPE(■)". *In: Proceedings 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning* (ADPRL-2007), Honolulu, US, pp. 338–345 (2007)

[20] C. Li , J. Zhang and Y. Li .ªpplication of artificial neural Network Based on Q-learning for mobile robot path planning" , *Proceedings in the 2006 IEEE International Conference on Information Acquisition* August 20-23 , 2006 Weihai , Shandong , China

[21] M.L. Littman "Markov games as a framework for multi-agent reinforcement learning."*Proceedings 11th International Conference on Machine Learning (ICML-1994)*, New Brunswick, US, pp. 157–163 (1994)

[22] C. Guestrin, M.G. Lagoudakis and R. Parr : Çoordinated reinforcement learning". *Proceedings 19th International Conference on Machine Learning (ICML-2002)*, Sydney, Australia, pp. 227–234 (2002)

[23] J.R. Kok, M.T.J. Spaan and N. Vlassis : "Non-communicative multi-robot coordination in dynamic environment". *Robotics and Autonomous Systems* 50(2–3), 99–114 (2005)

[24] C. Claus, C. Boutilier : "The dynamics of reinforcement learning in cooperative multiagent systems". *Proceedings 15th National Conference on Artificial Intelligence and 10th Conference on Innovative Applications of Artificial Intelligence* (AAAI/IAAI-1998), Madison, US, pp. 746–752 (1998)

[25] X. Wang, T. Sandholm : Reinforcement learning to play an optimal Nash equilibrium in team Markov games ". *Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems 15*, pp. 1571–1578. MIT Press, Cambridge (2003)

[26] N. Vlassis : .ª Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence". *Synthesis Lectures in Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers (2007)

[27] M.T.J. Spaan, N. Vlassis, F.C.A. Groen "High level coordination of agents based on multiagent Markov decision processes with roles". *Workshop on Cooperative Robotics*, 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2002), Lausanne, Switzerland, pp. 66–73 (2002)

[28] F. Fischer, M. Rovatsos and G. Weiss "Hierarchical reinforcement learning in communication-mediated multiagent coordination"*Proceedings 3rd International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS-2004), New York, US, pp. 1334–1335 (2004)

[29] M. Tan : "Multi-agent reinforcement learning: Independent vs. cooperative agents"*Proceedings 10th International Conference on Machine Learning* (ICML 1993), Amherst, US, pp. 330–337 (1993)

[30] C. Boutilier : "Planning, learning and coordination in multiagent decision processes". *Proceedings 6th Conference on Theoretical Aspects of Rationality and Knowledge* (TARK- 1996), pp. 195–210. De Zeeuwse Stromen, The Netherlands (1996)

[31] M.V. Nagendra Prasad, V.R. Lesser and S.E. Lander : "Learning organizational roles for negotiated search in a multiagent system". *International Journal of Human-Computer Studies* 48(1), 51–67 (1998)

[32] J.R. Kok, P.J. 't Hoen, B. Bakker and N.Vlassis : Ütile coordination: Learning interdependencies among cooperative agents". *Proceedings IEEE Symposium on Computational Intelligence and Games* (CIG 2005), Colchester, United Kingdom, pp. 29–36 (2005)

[33] M.L. Littman and P. Stone : "Implicit negotiation in repeated games". Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS (LNAI), vol. 2333, pp. 96–105. Springer, Heidelberg (2002)

[34] S. Nash, A. Sofer , *Linear and Nonlinear Programming.* McGraw-Hill, New York (1996)

[35] A. Merke and M.A. Riedmiller Reinforcement learning approach to robotic soccer". Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, pp. 435–440. Springer, Heidelberg (2002)

[36] T. Basar and G.J Olsder : "Dynamic Noncooperative Game Theory", 2nd edn. Society for Industrial and Applied Mathematics, SIAM (1999)

[37] A. Greenwald and A. Hall K: Çorrelated-Q learning". *Proceedings 20th International Conference on Machine Learning* (ICML-2003), Washington, US, pp. 242–249 (2003)

[38] M. Bowling and M. Veloso : Rational and convergent learning in stochastic games". *Proceedings 17th International Conference on Artificial Intelligence* (IJCAI-2001), San Francisco, US, pp. 1021–1026 (2001)

[39] M. Bowling and M. Veloso : .An analysis of stochastic game theory for multiagent reinforcement learning."*Tech. rep., Computer Science Dept.*, CarnegieMellon University, Pittsburgh, US (2000)

[40] J. Hu and M.P. Wellman : "Nash Q-learning for general-sum stochastic games". *Journal of Machine Learning Research* 4, 1039–1069 (2003)

[41] M. Lauer and M. Riedmiller : .An algorithm for distributed reinforcement learning in cooperative multi-agent systems". *Proceedings 17th International Conference on Machine Learning* (ICML-2000), Stanford University, US, pp. 535–542 (2000)

[42] R.H. Crites and A.G. Barto : "Improving elevator performance using reinforcement learning."Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (eds.) Advances in Neural Information Processing Systems 8, pp. 1017–1023. MIT Press, Cambridge (1996)

[43] L. Busoniu, R. Babuska and B. De Schutter : .A comprehensive survey of multi-agent reinforcement learning". *IEEE Transactions on Systems, Man, and Cybernetics. Part C: Applications and Reviews* 38(2), 156–172 (2008)

[44] M.J. Mataric: Reward functions for accelerated learning."*Proceedings 11th International Conference on Machine Learning* (ICML-1994), New Brunswick, US, pp. 181–189 (1994)

[45] M. Wooldridge , .An Introduction to MultiAgent Systems", John Wiley & Sons, 2002

[46] S. Bhattacharya, M. Likhachev and V. Kumar, "Multi-agent Path Planning with Multiple Tasks and Distance Constraints", *IEEE International Conference on Robotics and Automation*, 953-959, Anchorage, Alaska, USA, 2010

[47] K-H. Wang and A. Botea, "MAPP: a Scalable Multi-Agent Path Planning Algorithm with Tractability and Completeness Guarantees", *Journal of Artificial Intelligence Research,* Vol.42, 55-90, 2011

[48] V. R. Desaraju and J. P. How "Decentralized Path Planning for Multi-Agent Teams in Complex Environments using Rapidly-exploring Random Trees", *IEEE International Conference on Robotics and Automation*, 4956-4962, Shanghai, China, 2011

[49] Z. Cai and Z. Peng, Çooperative Coevolutionary Adaptive Genetic Algorithm in Path Planning of Cooperative Multi-Mobile Robot Systems", *Journal of Intelligent and Robotic Systems,* , Vol.33, No.1, 61-71, 2002

[50] N. Kwak, S. Ji, B. Lee, .A knowledge base for dynamic path planning of multi-agents", *16th IFAC World Congress,* 1363-1363, Czech Republic, 2005

[51] B. Bakker, M. Steingrover , R. Schouten, E. Nijhuis and L. Kester : Çooperative multi-agent reinforcement learning of traffic lights". *Workshop on Cooperative Multi-Agent Learning*, 16th European Conference on Machine Learning (ECML-2005), Porto, Portugal (2005)

[52] R.H. Crites, A.G. Barto : .Elevator group control using multiple reinforcement learning agents". *Machine Learning* 33(2–3), 235–262 (1998)

[53] H.V.D Parunak .: "Industrial and practical applications of DAI". Weiss, G. (ed.) Multi–Agent Systems: A Modern Approach to Distributed Artificial Intelligence, ch. 9, pp. 377–412. MIT Press, Cambridge (1999)

[54] S. Haykin, Neural Networks A Comprehensive Foundation, Second edition Prentice Hall International Inc

[55] L. Busoniu, R. Babuska and B. De Schutter, "Multi-agent Reinforcement Learning: An Overview", Innovations in Multi-Agent Systems and Applications - 1 Volume 310 of the series Studies in Computational Intelligence pp 183-221

[56] J.M. Vidal : "Learning in multiagent systems: An introduction from a game-theoretic perspective". Alonso, E., Kudenko, D., Kazakov, D. (eds.) AAMAS 2000 and AAMAS 2002. LNCS (LNAI), vol. 2636, pp. 202–215. Springer, Heidelberg (2003)

[57] E. A. Nadaraya, .ºn Estimating Regression", *Theory of Probability and its Applications,* Vol.9, No.1, 141-142, 1964

[58] M. B. Priestley and M. T. Chao, "Non-parametric function fitting," *J. Royal Statistical Soc., Ser. B*, vol. 34, pp. 385–392, 1972

[59] http://www.k-team.com/, *K-Team Corporation,* 2013

[60] S. C. Yun, S. Parasuraman and V, Ganapathy "Mobile Robot Navigation: Neural Q-learning" , *Advances in Computing & Inf. Technology* ,AISC 178, pp 259-268

[61] V. Ganapathy, S. C. Yun adn W. L. D. Lui Ütilization of webots and Khepera II as a Platform for neural Q-learning controllers", *2009 IEEE Symposium on Industrial Electronics and Applications* (ISIEA 2009) Octuber 4-6 Kuala Lumpur , Malaysia.

[62] The Mathworks Inc 2015

[63] Hu X. , and Wang Y. , Çonsensus of Linear Multi-Agent Systems Subject to Actuator Saturation ", *International Journal of Control, Automation, and Systems*, Vol.11, No.4, 649-656, 2013

[64] Luviano D. and Yu W. "Multi-agent path planning in unknown environment with reinforcement learning and neural network", *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* Octuber 2014 San Diego California USA

[65] Abul O. Polat F. and Alhajj R. "Multi-agent reinforcement learning using function approximation".*IEEE transactions on Systems, Man and Cybernetics Part C: Applications and reviews* 485-497 2000

[66] Luviano D. and Yu W. "Path planning in unknown environment with kernel smoothing and reinforcement learning for multi-agent systems"12th *International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 2015 octuber

[67] Fernandez F. , Parker L.E. "Learning in large cooperative multi-robots systems "*International Journal of Robotics and Automatization , Special Issue on Computational Intelligence Techniques in Cooperative Robots* , 16(4) , 217-226 2001.

[68] Tamakoshi H., Ishi S. "Multi agent reinforcement learning applied to a chase problem in a continuos world"*Artifitial Life and Robotics 202-206 2001*

[69] Harsanyi J.C and Selten R. " A General Theory of Equilibrium Selection in Games". MIT Press, Cambridge, 1988

[70] Busoniu L. , De Schutter B. and Babuska R. "Decentralized Reinforcement Learning Control of a robotic Manipulator", *International Conference on Control, Automation, Robotics and Vision, 2006. I CARCV '06. 9th*

[71] Bertsekas D.P. "Dynamic Programming and optimal control vol 2 ", third edition , Athena Scientific

[72] Istratesku V. "Fixed Point Theory : An introduction "Springer 2002

[73] Melo F.S. , Meyn S.P and Ribeiro M.I .An analysis of reinforcement learning with functions approximation", *Proceedings 25th International Conference on Machine Learning (ICML-08)* , Helsinky, Finland ,5-9 july 2008 pp- 664-671

[74] Szepesvari Cs. and Smart W.D "Interpolation baes Q-learning" *Procedings 21st International Conference on Machine Learning (ICML-04)* , Bannf Canada pp-791-798

[75] Bertsekas, D. P., and Tsitsiklis, J. N. (1996). "Neuro-dynamic programming ". Athena Scientific.

[76] Tsitsiklis, J. N. and Van Roy, B. "Feature-based methods for large scale dynamic programming ". *Machine Learning*, 22(1–3):59–94 1996

[77] R. Kruse, , Gebhardt, J. E., and Klowon, F. "Foundations of Fuzzy Systems ". Wiley 1994

[78] G. J. Gordon, Reinforcement learning with function approximation converges to a region ". In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 1040–1046. MIT Press. 2001

[79] Berenji, H. R. and Khedkar, P. "Learning and tuning fuzzy logic controllers through reinforcements". *IEEE Transactions on Neural Networks*, 3(5):724–740 1992

[80] Munos, R. and Moore, A."Variable-resolution discretization in optimal control ". *Machine Learning* , 49 (2–3) : 291–323 2002

[81] Chow, C.-S. and Tsitsiklis, J. N..An optimal one-way multigrid algorithm for discrete-time stochastic control ". *IEEE Transactions on Automatic Control*, 36 (8):898–914 1991

[82] Kaya M.,Alhajj R. "Modular fuzzy-reinforcement learning approach with internal model capabilities for multiagent systems"

[83] Choi Y. C., Ahn H. S., .A survey on multiagent reinforcement learning: coordination problems", 2010 IEEE

[84] Chen G., Cao W., Chen X.,Wu M, : "Multi agent Q learning with Joint state value approximation", *Procedings of the 30th Chines control conference* July 22-24 2011, Yantai China

[85] Wu M., Cao W., Peng J., She J., Chen X., "Balanced reactive-deliberative architecture for multi agent system for simulation league of Robo Cup ", *International journal of control, Automation and systems* 7(6) 945-955 ,2009

[86] Ren W., Beard R., .A decetralized scheme for spacecraft formation flying via virtual structure approach ", *Journal Guid. Control Dynamics* Vol 27, no.1 pp-73-82 Jan/Feb 2004

[87] Fax J. A., Murray R., "Information flow and cooperative control of vehicle formation". *IEEE transactions on automatic control* Vol.49 no.9 1465,1476 2004

[88] Kudenko D., KAzakov , Alonso E., .Adaptive agents and Multiagents systems", Springer Verlag 2005.

[89] Lynch N.A. Distributed Algorithms . San francisco California : Morgan Kauffman 1997

[90] Bertsekas D., Tsitsiklis J. Computacion paralelo y distribuida , Upper Saddle river NJ. Prentice Hall 1989

[91] Mataric M.J. , Reinforcemen learning in the multi-robot domain", Auto Robots. vol.4 No.1 pp73-83 1997

[92] Stephan V. , Debes K., Gross H., .A reinforcemen learning based neural multiagent system for control of a combustion process ", *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Netw.* (IJCNN-00) Italy

[93] Arel I., Liu C., Urbanik T., Kohls A.G., : Reinforcement learning-based multi-agent system for network traffic signal control ", *IEEE Inteligent transport system 2010*

[94] Riedmiller M., Moore A., Schneider J., Reinforcement Learning for Cooperating and Communicating Reactive Agents in Electrical Power Grids", *Balancing reactivity and social deliberation in Multiagent systems* New York Springer 2000

[95] Szer D., s Charpillet F., "Improving Coordination with Communication in Multi-agent Reinforcement learning", *Proceedings of the 16th IEEE International Conference on Tools with Artificial*

[96] Dinnissen P., Givigi S., Schwartz H., "Map Merging of Multi-Robot SLAM using Reinforcement Learning", 2012 *IEEE International Conference on Systems, Man, and Cybernetics* , Korea *Intelligence* (ICTAI 2004)

[97] Wang Y., de Silva C., .An Object Transportation System with Multiple Robots and Machine Learning",2005 *American Control Conference Potland* USA

[98] Zimmermann H., Neuneir R., Grothmann F., "Multiagent modelin of multiple Fx-Markets by neural networks", *IEEE transactions on neuran networks* Vol. 12 No.4 2001

[99] Huang J., Yang B. and Liu B.T. .A Distributed Q-Learning Algorithm for Multi-Agent Team Coordination", *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou*, 18-21 August 2005