UACJ INVESTIGACIÓN

Título del Proyecto de Investigación a que corresponde el Reporte Técnico:

Planeación de trayectorias para robots móviles en entornos desconocidos mediante aprendizaje por reforzamiento y control inteligente

Tipo de financiamiento

Financiado administrado por la UACJ

Autores del reporte técnico:

Dr. David Luviano Cruz
Dr. Luis Asuncion Perez Dominguez
Dr. Luis Carlos Mendez Gonzalez
M.C Francesco Garcia Luna

Planeación de trayectorias para robots móviles en entornos desconocidos mediante aprendizaje por reforzamiento y control inteligente

Resumen del reporte técnico en español (máximo 250 palabras)

En la industria y la manufactura se encuentra presenta la necesidad de contar con entidades autónomas capaces de desarrollar actividades encomendadas con una mínima supervisión y con la capacidad de aprender mediante la interacción con el entorno de trabajo. Específicamente, en el área de la robótica móvil, un campo abierto de investigación es la navegación autónoma mediante la generación de trayectorias en entornos desconocidos o inciertos para el agente, la cual consiste en generar un camino desde un punto inicial hasta un punto final bajo ciertas restricciones tales como: evitar colisión con obstáculos y otros robots, coordinarse entre ellos, cumplir una cierta tarea optimizando algún parámetro en específico como tiempo, energía o costo. Los objetivos que se persiguen con el presente proyecto son: Desarrollar un algoritmo de aprendizaje por reforzamiento iterativo determinístico el cual será complementado con una red neuronal multicapa, la cual ofrecerá información aproximada acerca de acciones sub optimas a ejecutar. Lo anterior con la finalidad de implementarlo experimentalmente en robots móviles para la planeación de trayectorias, evitando colisiones con obstáculos y otros robots presentes en el espacio de trabajo, hasta cumplir su objetivo final encomendado.

Resumen del reporte técnico en inglés (máximo 250alabras):

In industry and manufacturing, there is a need for autonomous entities capable of developing mandated activities with minimal supervision and with the ability to learn through interaction with the work environment. Specifically, in the area of mobile robotics, an open field of research is autonomous navigation through the generation of trajectories in unknown or uncertain environments for the agent, which consists of generating a path from an initial point to an end point under certain restrictions such as: avoid collision with obstacles and other robots, coordinate with each other, fulfill a certain task optimizing a specific parameter such as time, energy or cost. The objectives pursued with this project are: To develop a deterministic iterative reinforcement learning algorithm which will be complemented with a multilayer neural network, which will offer approximate information about suboptimal actions to execute. The above in order to implement it experimentally in mobile robots for trajectory planning, avoiding collisions with obstacles and other robots present in the workspace, until you meet your final entrusted goal.

Palabras clave:

Multi agentes, reinforcement learning, neural networks

Usuarios potenciales (del proyecto de investigación)

Automóviles autónomos, industria de la manufactura y logística.

Reconocimientos (agradecimientos a la institución, estudiantes que colaboraron, instituciones que apoyaron a la realización del proyecto, etc.)

Agradecemos a la Universidad autónoma de ciudad Juárez (UACJ) por proporcionar las instalaciones e infraestructura para el correcto desarrollo del proyecto, a la secretaria de educación pública (SEP) por la aportación de los fondos por medio del Apoyo a la Incorporación de NPTC (PRODEP) acuerdo 511-6 / 17-7605, extendemos el agradecimiento a los estudiantes de ingeniería mecatrónica: Pedro Hurtado y Miguel Angel Rodriguez Ruiz por sus aportaciones en el proyecto.

1. INTRODUCCIÓN

Actualmente la industria y la manufactura se encuentran en la necesitad de dispositivos y aparatos que puedan adaptarse a las condiciones cambiantes en las configuraciones de espacio presentes en los ambientes de trabajo, por lo que entidades preprogramadas pueden ser consideradas como no aptas a desempeñarse en medios cambiantes. En este sentido es necesario contar con entidades, las cuales puedan aprender a realizar una actividad con solo la interacción con el medio a través de prueba y error. La teoría de aprendizaje por reforzamiento es adecuada para este tipo de problemática en donde no se cuenta con un modelo del entorno de trabajo, ya que permite el aprendizaje a través de la experimentación y de una señal de recompensa.

En específico, la navegación en robótica móvil ofrece una amplia oportunidad de investigación debido a los retos que ofrece en el campo de sensores, actuadores, visión, autonomía. En particular, un tema abierto en esta área es el aprendizaje del agente (robot) de las acciones a seguir para lograr un objetivo preestablecido o para generar una trayectoria, únicamente por medio de la interacción del entorno de trabajo con el robot. De tal manera, que no sea necesario un comportamiento preprogramado del robot móvil, si no que mediante la experiencia se logre el aprendizaje mediante prueba y error**.**

2. PLANTEAMIENTO

Diversos tipos de funciones de aproximación han sido propuestos en el campo de la teoría de control y del aprendizaje máquina, entre ellos, las redes neuronales ofrecen la particularidad de realizar aproximaciones de funciones no lineales, siempre y cuando se tenga acceso a una serie de datos de entrenamiento adecuados [1]. En lo que refiere tareas en que involucran sistemas no lineales o sistemas en donde una programación previa para resolver un problema no es posible, el aprendizaje por reforzamiento (RL) ha sido utilizado con éxito [2], ya que permite a los agentes aprender nuevos comportamientos de tal manera que puedan predecir y adaptarse a los cambios en el medio en donde se desarrollan [3].

El aprendizaje por reforzamiento es un método semi supervisado de aprendizaje, en donde el objetivo es maximizar una función de recompensa escalar la cual es definida por el entorno y las entidades que interactúan con este, llamados agentes. En cada paso de aprendizaje, el agente toma una medición del entorno y toma una acción, lo que motiva que el medio transite a un nuevo estado, esta transición es evaluada por medio de la función de recompensa escalar, es importante mencionar, que a los agentes no se les dice que acción tomar, por lo que deben de explorar el entorno para encontrar las acciones que le proporcionen una mayor recompensa.

Distintos tipos de aplicaciones del aprendizaje por reforzamiento han tenido cabida en la teoría control, entre ellos, para manejar y controlar una red inteligente eléctrica [4], por medio de visión y RL enseñar a un robot a realizar disparos de una pelota a una meta [5], utilizar RL profundo para que un brazo manipulador aprenda movimientos en 3D [6], para coordinar actividades cooperativas entre agentes [7].

Un área en donde el aprendizaje por reforzamiento ha sido provechoso es la planeación de trayectoria para robots móviles, en donde se genera una trayectoria desde un punto de inicio hasta un punto final respetando ciertas restricciones impuestas al desplazamiento, tales como obstáculos, delimitación del área de trayectoria. La generación de trayectoria puede ser realizada con distintos tipos de algoritmos de búsqueda, en [8] el método de grafos es utilizado para diseñar un espacio de tareas mediante búsqueda heurística, en [9] gráficos un direccionados son usados para la planeación de trayectoria, en [10] utilizan algoritmos genéticos para encontrar una trayectoria óptima.

La generalización del proceso de decisión de Markov en el aprendizaje por reforzamiento en sistemas multi-agente (MARL) es el llamado juego estocástico [8]. El juego estocástico se define como una tupla $(S, A_1, A_2, ..., A_n, f, \rho_1, \rho_2, ..., \rho_n)$:

$$\rho_i : S \times \mathbf{A} \times S \to \mathbb{R}$$
$$f : S \times \mathbf{A} \times S \to [0,1]$$
$$\mathbf{A} = A_1 \times A_2 \ldots \times A_n$$

donde n es el número 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 los MAS 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 de la política conjunta:

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

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

$$Q_i^{\boldsymbol{\pi}} : S \times \mathbf{A} \to \mathrm{R}$$
$$Q_i^{\boldsymbol{\pi}}(s, \mathbf{a}) = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{i,k+1} \middle| s_0 = s, \mathbf{a}_0 = \mathbf{a}, \boldsymbol{\pi}\right\}$$

En juegos estocásticos 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.

La función Q óptima es definida como Q^*

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

La cual satisface la ecuación de optimalidad de Bellman:

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

Una vez que Q^* está disponible, una política de acciones óptima puede ser calculada por medio de escoger en cada estado una acción con el más grande valor Q óptimo:

$$\pi^*(x) = \arg\max_a Q^*(s,a)$$

Un enfoque generalizado utilizado para resolver el problema de la coordinación en MAS es asegurarse que cualquier situación de decisión sea resuelta de la misma manera por todos los agentes usando algún tipo de negociación. En nuestra propuesta se utiliza coordinación implícita, en donde los agentes aprenden a escoger acciones de manera coordinada por medio de prueba y error.

Los algoritmos que utilizan RL 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 número 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 número de estados y que el algoritmo Q-Learning se basa en tablas de búsqueda, es usado el método de aproximación no paramétrica basado en Kernel para aproximar los estados desconocidos que no son logrados visitar cuando el algoritmo RL es llevado a cabo, 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:

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

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 preespecificado t. El Kernel es simplemente un promedio ponderado de todos los puntos de datos:

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

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 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:

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

La regresión llevada a cabo por el Kernel para los datos $[a(k), s(t)]$ de acuerdo a Nadaraya-Watson [10], donde $t \in [\tau_1, \tau_2]$ es:

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

En las aplicaciones en control automático todas las variables aleatorias tienen 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:

$$K_t[s(t)] \to \frac{N}{\tau_2}$$

Por lo tanto, la regresión propuesta por Nadaraya usado en el kernel puede ser reemplazada por la regresión propuesta por Priestley-Chao [10], el cual es definida como:

$$\hat{\phi}(s(t)) = \frac{\tau_2}{N}\sum_{t=\tau_1}^{\tau_2} K_t[s(t)]a(t) \qquad (1)$$

Dado lo anterior la expresión (1) es usado para modelar los estados desconocidos o que aún no han sido explorados del entorno. Una forma posible es diseñar una función kernel amplía con el fin de abarcar una mayor cantidad de datos por medio de una amplitud completa:

$$\sigma_{\max} = 2\sqrt{2\ln(2)}\sigma$$

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

$$\sigma = \frac{N}{20\sqrt{2\ln(2)}}$$

donde N es el total de datos disponibles.

3. METODOLOGÍA

El flujo de aprendizaje es mostrado en la Figura 1.

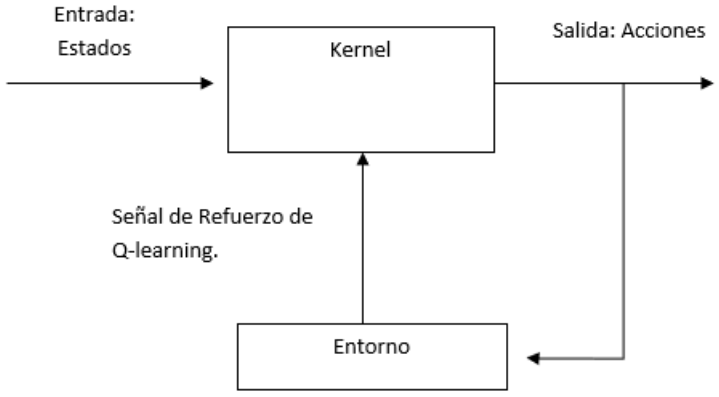

Figura 1. Flujo de aprendizaje propuesto.

Fuente: Autores

En cada paso de tiempo discreto t, los estados donde se encuentran los agentes son observados y estos son referidos a una tabla de estados-acciones llamada Q -tabla. El algoritmo Q-learning es usado para 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 porque 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 Kernel. La nueva aproximación será enviada a los agentes con la finalidad de continuar la tarea encomendada.

El aproximador previamente entrenado genera como salida una acción sub-óptima para cada agente en el entorno, de esta manera se evita volver ejecutar el algoritmo RL 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 RL Q-learning**: el estado actual del estado de los agentes con respeto al entorno es 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 más 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, los valores obtenidos son guardados en una tabla de búsqueda llamada Q- tabla.

5) **Repetir los pasos 2-4 hasta que los agentes alcance el objetivo planteado**: El ciclo de entrenamiento finaliza si los agentes alcanzan a cumplir el objetivo final planteado o se alcanza un límite de iteraciones establecido.

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 del Kernel**: Se usa la tabla de valores Q obtenida por el algoritmo Q-Learning para entrenar el kernel, cada columna de la tabla Q representa un estado, la cual es introducido como entrada y las acciones óptimas encontradas como salidas del sistema.

9) **Obtención de las acciones óptimas aproximadas:** Una vez entrenado el kernel, este 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.

4. RESULTADOS

Con la finalidad de validar el desempeño del método propuesto, se usan dos robots móviles Khepera IV, cuyo objetivo es generar una trayectoria desde un punto inicial hasta una meta. El software utilizado fue Matlab, los robots fueron usados en modo esclavo con una conexión por medio de bluetooth a 115200 baudios, el intercambio de información entre los robots Khepera y Matlab fue mediante código ASCII. La tarea debe de completarse en un tiempo mínimo evitando obstáculos y coordinándose entre ellos, se asume que los agentes no tienen un conocimiento previo acerca de la posición y forma de los obstáculos presentes en el entorno, la configuración del entorno de trabajo se muestra en la Figura 2.

Figura 2. Configuración de la tarea.

Fuente: Autores

La posición inicial de los agentes se selecciona aleatoriamente y se lleva a cabo 50 pasos de aprendizaje, en el caso de alcanzar este límite, 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 lo que se detendrá la etapa de aprendizaje por reforzamiento.
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.

La función de recompensa $\rho(x,u)$ está dada por:

$$r_{k+1} = 1 \text{ si el agente toma el objetivo}$$

$$r_{k+1} = 10 \text{ si los agentes llevan el objetivo hasta la posición base}$$

$$r_{k+1} = 0 \text{ en cualquier otra situación}$$

La función de recompensa es diseñada de tal manera que al asignar el valor numérico 1 cuando el agente toma el objetivo se garantiza que se conserve la propiedad de Márkov, la recompensa numérica de 10 para cuando el agente alcanza el objetivo final evita que los agentes encuentren estados subóptimos motivados por recompensas intermedias.

La convergencia del algoritmo es mostrada en la Figura 3. En esta figura se muestra que el algoritmo converge después de 16 ensayos, el tiempo de duración de cada trial depende de la cantidad de pasos que los agentes realizan antes de detener el ensayo o de alcanzar el objetivo de aprendizaje.

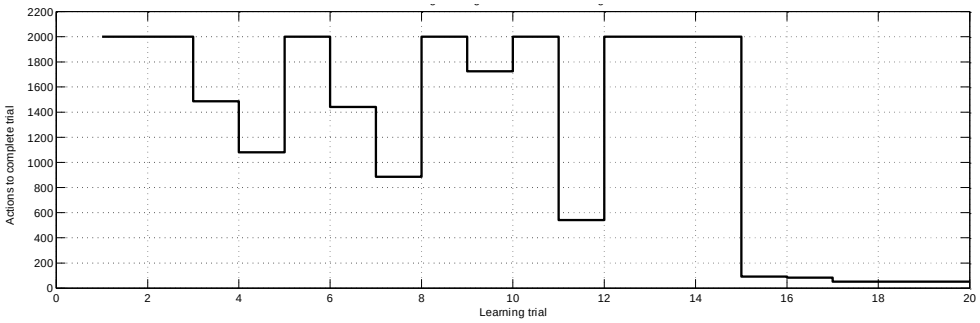

Figura 3. Convergencia de algoritmo propuesto

Fuente: Autores

El conjunto de datos que serán empleados como muestras de entrenamiento para el Kernel son tomados de la Q-tabla óptima generada por el algoritmo Q-learning, Tabla I. Cada columna de la Q-tabla representa un estado y la salida del aproximador será la acción conjunta que los agentes deberán ejecutar.

	Recompensas dependiendo la acción					
Posición	Left	Right	Up	Down	Y Max	Mejor Acción
Q11	0.49027057	7.130905021	7.130905021	0.49027057	7.130905021	UP & RIGHT
Q12	3.300203452	9.968480681	2.130905488	5.490270104	9.968480681	RIGHT
Q13	0.49027057	0.49027057	0.49027057	0.49027057	0.49027057	N/A
Q14	7.292940763	13.77215438	10.68214364	7.292940763	13.77215438	RIGHT
Q15	5.947573542	12.31807983	5.682144102	12.2929403	12.31807983	RIGHT
Q21	5.490270104	9.968480681	9.968480681	3.300203452	9.968480681	UP & RIGHT
Q22	8.300202985	5.559819669	12.08929957	8.300202985	12.08929957	UP
Q23	7.292940763	13.77215438	13.77215438	10.5598192	13.77215438	UP & RIGHT
Q24	12.2929403	15.53603974	12.31807983	12.2929403	15.53603974	RIGHT
Q25	10.94757308	7.318080298	7.318080298	13.90550713	13.90550713	DOWN
Q31	8.300202985	12.08929957	4.968481147	5.559819669	12.08929957	RIGHT
Q32	0.49027057	0.49027057	0.49027057	0.49027057	0.49027057	N/A
Q33	12.2929403	15.53603974	15.53603974	8.905507598	15.53603974	UP & RIGHT
Q34	13.90550713	18.05877027	10.5360402	13.90550713	18.05877027	RIGHT
Q35	0.49027057	0.49027057	0.49027057	0.49027057	0.49027057	N/A
Q41	10.5598192	10.73914234	13.77215438	7.292940763	13.77215438	UP
Q42	8.905507598	12.34110216	15.53603974	12.2929403	15.53603974	UP
Q43	13.90550713	13.91672277	18.05877027	13.90550713	18.05877027	UP
Q44	15.97521583	14.0360657	21.82664072	15.97521583	21.82664072	UP
Q45	15.49026917	15.49026917	15.49026917	15.49026917	15.49026917	ALL
Q51	12.2929403	5.981495688	12.34110216	5.981495688	12.34110216	UP
Q52	13.90550713	7.459822216	13.91672277	10.98149522	13.91672277	UP
Q53	15.97521583	8.916723241	8.916723241	12.45982175	15.97521583	LEFT
Q54	0.49027057	0.49027057	0.49027057	0.49027057	0.49027057	N/A
Q55	23.14384188	14.10045021	14.10045021	14.10045021	23.14384188	LEFT

Tabla I. Q-tabla con acciones optimas

Con la finalidad de comparar el desempeño del algoritmo, se escoge como posición inicial para los agentes, un estado desconocido el cual no se encuentra en la Q-tabla, bajo esta situación no sería posible proveer las acciones óptimas a los agentes, por lo que el kernel ofrecerá una acción sub optima coordinada para cada agente, ejemplo de estas situaciones se muestran en las Figuras 4 y 5, donde los agentes tienen posición inicial en estados que no se encuentran en la Q-tabla.

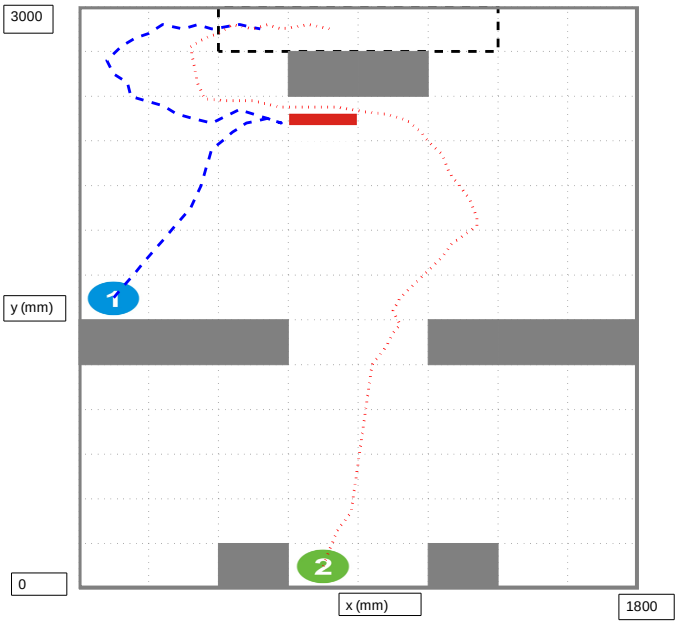

Figura 4. Primera trayectoria encontrada por el Kernel partiendo desde un estado desconocido.

Fuente: Autores

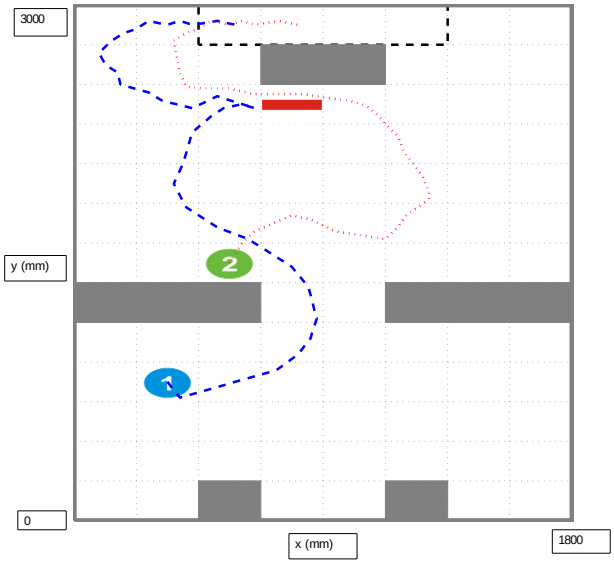

Figura 5 Segunda trayectoria encontrada por el Kernel partiendo desde un estado desconocido

Fuente: Autores

El vector de estado $x=[x_{1,}x_2,x_3,x_4]^T$ y la señal de control $u=[u_1,u_2]$ del agente 1 es mostrados en la Figura 6, donde x_1 es la posición en el x, x_2 es la posición en el eje y, x_3 es la velocidad en el eje x, x_4 es la velocidad en el eje y. u_1 es la fuerza aplicado en el eje x, u_2 es la fuerza aplicada en el eje y.

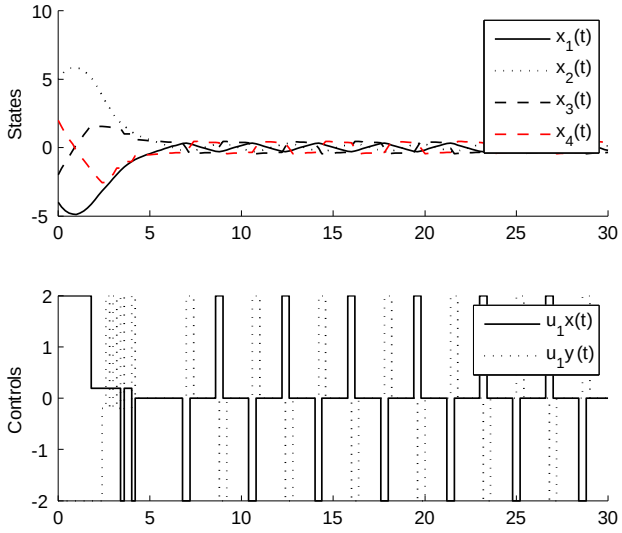

Figura 6. Estado y señal de control para el agente 1
Fuente: Autores

El vector de estado $x=[x_{1,}x_2,x_3,x_4]^T$ y la señal de control $u=[u_1,u_2]$ del agente 2 es mostrados en la Figura 7.

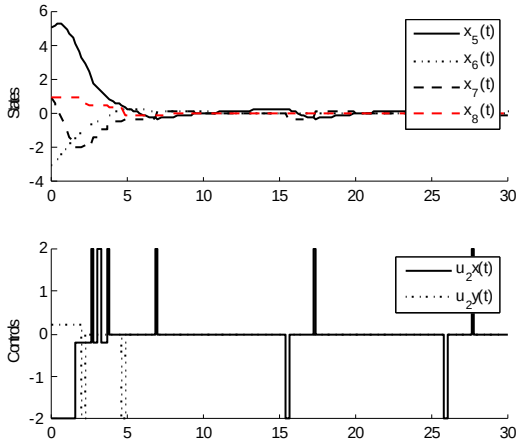

Figura 7. Estado y señal de control para el agente 1

5. CONCLUSIONES

En el presente artículo se presentó una propuesta de integración entre 2 estrategias de control. En una primera etapa se usa MARL como un medio para obtener datos e información de la tarea y del entorno en donde se desenvuelven los agentes, posteriormente estos datos son utilizados para entrenar un aproximador no paramétrico.

Con los resultados experimentales obtenidos se confirma 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, lo que lleva a un ahorro tiempo y poder computacional.

Es de notar que al utilizar un aproximador no paramétrico, el número de los pesos a sintonizar en el kernel aumentan conforme al tamaño de los datos disponibles para el entrenamiento crecen, por lo que se debe de buscar un balance entre simplicidad y precisión del aproximador.

Además, 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. El número mínimo de datos capturados por el algoritmo debe de ser suficientes para que estos describan la dinámica del sistema y junto con el diseño de una función de recompensa adecuada garantizar que exista un máximo local en la función de retorno R. La elección de los datos capturados dependerá del conocimiento previo que se tenga del problema a solucionar. La natural dirección en el estudio de este tema seria buscar técnicas donde se pudiera lidiar con sistemas multi agentes con estados continuos.

REFERENCIAS **(bibliografía)**

[1] Sofge D, White D (1990) Neural network-based process optimization and control. In Proceedings of the 29th IEEE Conference on Decision and Control. Hawai, USA.

[2] Kaelbling L. P., Reinforcement learning: A survey, Journal of Artificial Intelligence Research, 4(3), 237-285,1996

[3] Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction (Vol. 1, No. 1). Cambridge: MIT press.

[4] Kara, E. C., Berges, M., Krogh, B., & Kar, S. (2012, November). Using smart devices for system-level management and control in the smart grid: A reinforcement learning framework. In Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on (pp. 85-90). IEEE.

[5] Asada, M., Noda, S., Tawaratsumida, S., & Hosoda, K. (1996). Purposive behavior acquisition for a real robot by vision-based reinforcement learning. Machine learning, 23(2), 279-303.

[6] Gu, S., Holly, E., Lillicrap, T., & Levine, S. (2017, May). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Robotics and Automation (ICRA), 2017 IEEE International Conference on (pp. 3389-3396). IEEE.

[7] Foerster, J., Assael, Y. M., de Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In Advances in Neural Information Processing Systems (pp. 2137-2145).

[8] Bhattacharya, S., Likhachev, M., & Kumar, V. (2010, May). Multi-agent path planning with multiple tasks and distance constraints. In Robotics and Automation (ICRA), 2010 IEEE International Conference on (pp. 953-959). IEEE.

[9] Wang, K. H. C., & Botea, A. (2011). MAPP: a scalable multi-agent path planning algorithm with tractability and completeness guarantees. Journal of Artificial Intelligence Research, 42, 55-90.

[10] Cai, Z., & Peng, Z. (2002). Cooperative coevolutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot systems. Journal of Intelligent & Robotic Systems, 33(1), 61-71.

ANEXOS

Sin anexos

Productos generados

Ruiz, M. A. R., Cruz, D. L., Gonzalez, L. C. M., & Domínguez, L. P. Robot Path planning using reinforcement learning and nonlinear approximation function (Planeación de trayectoria utilizando aprendizaje por reforzamiento y función de aproximación).

Cruz, D. L., Luna, F. J. G., & Domínguez, L. A. P. (2018). Multiagent reinforcement learning using Non-Parametric Approximation. *Respuestas*, *23*(2), 53-61.

Luviano-Cruz, D., Garcia-Luna, F., Pérez-Domínguez, L., & Gadi, S. K. (2018). Multi-agent reinforcement learning using linear fuzzy model applied to cooperative mobile robots. *Symmetry*, *10*(10), 461.