Universidad Zaragoza
1542

Trabajo Fin de Grado

Planificación de trayectorias de sistemas multi-robot en entornos desconocidos
Path planning in unknown environments in multi-robot systems

Autor/es

Mathias Saury Echagüe

Director/es

Cristian Mahulea
Eduardo Montijano Muñoz

Escuela de Ingeniería y Arquitectura
2019

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

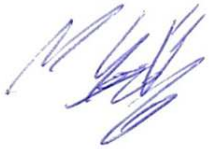
D./Dª. Mathias Saury Echagüe,

con nº de DNI X5315393R en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Planificación de trayectorias de sistemas multi-robot en entornos desconocidos,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 31 de Enero de 2019



Fdo: Mathias Saury Echagüe

Planificación de trayectorias de sistemas multi-robot en entornos desconocidos

Resumen

En la actualidad, uno de los problemas que se está abordando en la robótica es dar mayor autonomía a los robots móviles, entendiendo como autonomía el proceso de poder interactuar con el entorno para poder realizar acciones que le permitan ejecutar su tarea.

La cuestión que se abordará se basa en un problema de planificación de trayectorias en entornos desconocidos, donde un conjunto de robots en unas respectivas posiciones iniciales deberán encontrar el camino para llegar a distintos destinos finales.

El objetivo de este trabajo de fin de grado, consiste en evaluar un algoritmo basado en aprendizaje por refuerzo junto con algoritmos de consenso distribuido para implementar un sistema multi-robot que permita conseguir este fin. El algoritmo empleado para su resolución es Dyna-Q, el cual se estudia primero para el caso de un solo robot, para después extenderlo al caso multi-robot. En este último caso, se desarrollan y analizan distintos protocolos de comunicación que ayuden a la consecución del objetivo. Se han realizado simulaciones para evaluar el comportamiento de los robots con los distintos protocolos, examinando además cómo actúan con un número distinto de robots y de tamaños de entorno.

Contenido

1. Introducción

1.1. Motivación

La robótica es una rama de la ingeniería que lleva años implementada en la industria, pero aún sigue siendo un campo en continuo proceso de desarrollo y mejora. Particularizando para este proyecto, se encuentra la robótica móvil, un campo relativamente nuevo que permite explorar entornos difíciles, inaccesibles u hostiles al ser humano. En concreto, este TFG se centra en el problema de *planificación de trayectorias* para equipos compuestos por varios robots. Aunque este problema ha sido estudiado, el enfoque llevado en este proyecto se especializa en entornos desconocidos, utilizando para su resolución *un algoritmo de aprendizaje por refuerzo junto con algoritmos de consenso distribuido*, siendo la combinación de ambas una aproximación al problema poco estudiada a nivel académico.

A grandes rasgos, la idea es que varios robots, dados unos destinos finales deseados, tienen que desplegarse para alcanzar dichos destinos, sin restricciones sobre qué robot debe ir a qué destino. Cada uno de ellos tiene un conocimiento parcial del entorno, pudiendo detectar obstáculos a medida que avanzan y comunicándose si es posible para ayudarse entre ellos. A continuación, se presenta una breve descripción de las dos técnicas que se van a emplear en el TFG para abordar el problema.

1.1.1. Aprendizaje por refuerzo

El **aprendizaje automático** (también conocido como *Machine Learning*) es una técnica utilizada en muchas aplicaciones, siendo una rama de la informática que está en proceso de evolución. Se podría dividir los tipos de aprendizaje en 3 grupos, entre los cuales están el *aprendizaje supervisado*, cuyo objetivo principal es establecer relaciones entre datos de entrada y salida (crear una función para predecir un valor dada una entrada); el *aprendizaje no supervisado*, basado en relacionar las características semejantes de un grupo de variables para agruparlas según distintos criterios; y el **aprendizaje por refuerzo (AR)**.

El aprendizaje por refuerzo permite a un robot aprender un comportamiento que no había sido definido por el programador, donde este descubre su entorno y las diferentes consecuencias de sus acciones a través de interacciones con él, aprendiendo de su propia experiencia, sin tener conocimientos, fines o efectos preestablecidos. Algunos ejemplos de aplicación son el uso para una exploración espacial más autónoma, el desarrollo de robots domésticos que se adapten a diferentes domicilios (Figura 1a), dar al coche autónomo menos dependencia de la cartografía (Figura 1b), etc [1].

a) Robot aspirador.

b) Coche autónomo.

Figura 1. Ejemplos de aprendizaje por refuerzo[1].

El AR se basa en el uso de un *sistema de recompensas*, donde con cada acción que realice obtendrá una recompensa u otra. Por ello, el aprendizaje por refuerzo con la sola interacción con el medio, puede llegar a ser poco efectivo. Consecuentemente, este método de aprendizaje es conveniente combinarlo, como en el caso de este proyecto, con el *aprendizaje por planificación*, el cual se basa en un *modelo interno* para realizar simulaciones. De los algoritmos de AR, se ha elegido utilizar **Dyna-Q**, un algoritmo que *combina aprendizaje por refuerzo y planificación*, debido a que en entornos desconocidos es adecuado, permitiendo exploración del entorno y la búsqueda de la trayectoria más corta, sin complicar en exceso su implementación para sistemas multi-robot.

1.1.2. Consenso distribuido

Un problema que ha surgido en el contexto de los *sistemas multiagente*[2] (SMA) y de los sistemas multi-robot es el convenio de grupo, donde cada robot puede tener una opinión inicial diferente sobre el mismo objetivo, por ejemplo, la descripción del mapa [2].

Con el fin de asegurar la coordinación entre los robots, cada robot comparte información con los robots más próximos para que todos puedan alcanzar un **consenso** sobre un objetivo de interés común, haciendo que manejen una información más homogénea. La solución de este tipo de problemas toma la forma de algoritmos o protocolos distribuidos, donde se busca garantizar que todos los robots acaban asignando el mismo valor a la información gestionada. Algunos ejemplos que utilizan esta tecnología son: el valor de ciertas medidas en una red de sensores, la dirección de formación de UAVs (vehículos aéreos no tripulados), o la posición de un objetivo para un grupo de robots de vigilancia [3].

---

[1] Imagen a), fuente: http://www.blauden.com/robot-aspirador-irobot-roomba-580

Imagen b), fuente: https://cincodias.elpais.com/cincodias/2015/05/15/motor/1431680158_975445.html

[2] Un **agente** es un sistema situado dentro de un entorno formando parte integrante de él, que tiene las propiedades de ser autónomo, tener capacidad de reacción, pro-actividad y sociabilidad. En el contexto de este trabajo se considera que un agente es un robot.

Mediante una interacción de cooperación[3] entre los robots, se trataría de compartir información (alcanzando un consenso) para obtener un conocimiento del entorno más completo para cada uno de ellos, mejorando con ello la obtención de una solución. Esto permite que un sistema multi-robot en el caso que se va a representar, por ejemplo, un robot haciendo uso de Dyna-Q tratando de llegar a un destino final, gracias a la información transmitida por un robot con previo conocimiento en el entorno, le haya comunicado que esa posición está ocupada. También comunicarle que el camino que está siguiendo le lleva directamente a un obstáculo, permitiendo que este corrija su trayectoria y aminorando el número de pasos que realiza.

1.2. Objetivos y alcance

El objetivo principal de este proyecto es el desarrollo de un algoritmo de planificación de trayectorias para equipos de robots en entornos desconocidos o parcialmente desconocidos, combinando técnicas de aprendizaje por refuerzo y consenso distribuido. Para conseguir este objetivo, en el TFG se plantean los siguientes sub-objetivos:

- Extender el algoritmo Dyna-Q a sistemas multi-robot con algoritmos de consenso distribuido.

- Desarrollar distintos protocolos de comunicación para comprobar las distintas opciones a la hora de transmitir información y su influencia en las trayectorias de cada robot.

- Analizar el comportamiento de los robots en distintos tipos de entorno, centrándose en la variación del tamaño del mapa y del número total de robots en el entorno.

Con respecto al alcance de los objetivos, el cual se irá desarrollando a lo largo de esta memoria, se destaca lo siguiente:

- Se ha estudiado de manera intensiva la teoría referente al algoritmo Dyna-Q, en concreto, se ha partido del trabajo de fin de grado [5], donde se implementó Dyna-Q, especialmente para el caso mono-robot y en caso multi-robot con una interacción apática[4] entre los robots.

- Se han propuesto y analizado diferentes protocolos de comunicación favoreciendo el intercambio de información entre robots vecinos, buscando con ello mejorar el comportamiento global del equipo.

[3] Comportamiento donde los diferentes robots tratan de ayudarse mutuamente a conseguir sus objetivos o un objetivo común.
[4] Comportamiento donde cada robot realiza su tarea independientemente de los objetivos que tengan los demás.

- Se han realizado simulaciones en *Matlab* por medio de la *toolbox RMtool* en el clúster *Hermes* de I3A para la obtención de datos y se han analizado los distintos resultados para la extracción de evaluaciones concluyentes.

1.3. Estructura

La forma en la que está estructurada esta memoria es la siguiente:

- En la sección 2, se da una descripción del problema a resolver en este proyecto.

- En la sección 3, se explica el algoritmo Dyna-Q para un robot, y sus distintos parámetros.

- En la sección 4, se extiende Dyna-Q al caso multi-robot, donde se proponen diferentes protocolos de comunicación.

- En la sección 5, se explican las herramientas utilizadas para la evaluación empírica de los métodos propuestos y se analizan los resultados obtenidos.

- Para terminar, en la sección 6 se exponen las conclusiones de este trabajo.

2. Descripción del problema

En esta sección, se explica el problema a resolver a lo largo del proyecto, que consiste en la planificación de trayectorias por un sistema multi-robot, con el objetivo de resolverlo.

Partiendo de un entorno 2D, se encuentra un conjunto de *N* robots móviles idénticos. Un conjunto de *B* obstáculos estáticos se ubica en el entorno (Figura 2) siendo imposible por los robots atravesarlos. Su existencia es desconocida por los robots hasta que los identifican, a través de un sensor que tienen equipado. Los robots se detectan entre sí como obstáculos dinámicos, pudiendo diferenciar entre obstáculo y robot. La posición de los obstáculos también puede ser conocida por un robot a través de un proceso de comunicación. La comunicación es distribuida y se produce entre robots vecinos si se encuentran a una distancia inferior a *R* (que se denominará como radio de comunicación). La comunicación no se ve afectada por los obstáculos, la posición del robot en el mapa (siendo en cada estado igual en términos de cobertura o calidad), o la distancia entre robots. Es decir, si dos robots se encuentran a una distancia inferior a *R*, la transmisión de información se produce de la misma forma independientemente de la distancia entre ellos.

Se tiene un entorno cerrado, dividido en un conjunto de estados $S = \{s_1, s_2, \ldots, s_{|S|}\}$, donde cada $s_k \in S$ es un estado. Los robots conocen la partición del entorno, pero desconocen el conjunto de regiones ocupadas por obstáculos, $O \subseteq S$. Cada robot conoce su posición inicial, $s_0 \in S$, y el conjunto de posiciones de los *N* destinos (igual al número de robots), $F \subseteq S$. El conjunto de acciones posibles, *A,* es conocido, al igual que la adyacencia entre estados. Utilizando la adyacencia entre estados, se construye una función de transición $f: S \times A \rightarrow S$. Todos los destinos se consideran alcanzables y ningún robot tiene inicialmente conocimiento alguno sobre si un destino ha sido alcanzado.

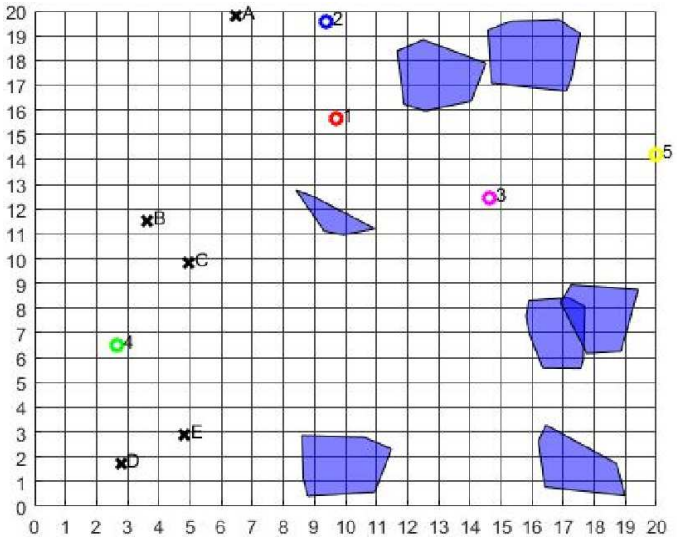


Figura 2. Entorno real (donde 1, 2, 3, 4, 5 son las posiciones iniciales, A, B, C, D, E son las posiciones finales y los polígonos azules los obstáculos).

El objetivo de cada robot es encontrar una **política,** $\pi(s)$: $S \rightarrow A$, que determine para cada estado $s \in S$ la mejor acción a realizar por el robot, con el fin de alcanzar uno de los N destinos. Esta política se evalúa con la **matriz de recompensas** Q, que dado un estado $s \in S$ y una acción $a \in A$, $Q(s,a)$ es la recompensa de ejecutar la acción a en el estado s. Cuanto más grande sea la recompensa, mejor es la acción correspondiente (acerca al robot a uno de los destinos). Todos estos datos componen el **modelo interno**, que se define como $M_i = (S, f_i, A, O_i, F_i, Q_i)$ de cada robot i, que se utilizará para generar experiencia simulada y mejorar la política $\pi(s)$. Cada robot ejecuta un algoritmo Dyna-Q, que mejora la política y dirige hacia uno de los destinos posibles no alcanzados por otros robots, con el objetivo del sistema multi-robot de que todos los N destinos sean alcanzados.

3. Planificación de la trayectoria de un robot

Con el objetivo de comprender la naturaleza del algoritmo Dyna-Q, esta sección se centra en los conceptos fundamentales de Dyna-Q y sus distintos parámetros, explicados para el caso de un único robot y destino. Se pretende que sirva como punto de partida para su extensión al caso multi-robot, debido a su semejanza con el caso mono-robot, facilitando la comprensión de los conceptos.

3.1. Descripción Dyna-Q

El marco de aprendizaje por refuerzo se basa en un aprendiz (robot), que interactúa con el medio comprendiendo todo lo que está a su alrededor. A partir de estas interacciones con el medio, se producen una serie de valores numéricos (llamados recompensas), que el robot analiza y en un proceso de toma de decisiones, alcanza el objetivo de la manera más efectiva posible, es decir, buscando maximizar las recompensas [4]. En un problema de planificación de trayectorias, esta interacción se basa en el movimiento del robot analizando el estado al que se ha desplazado.

Es importante entender este proceso de toma de decisiones, ya que es el fundamento central del comportamiento del robot a la hora de elegir sus acciones dado un determinado estado, y por lo tanto de ayudarle a obtener un modelo más cercano al entorno real y la obtención de una trayectoria satisfactoria. En el entorno de trabajo del AR, el robot, a través de una política que determina su comportamiento en un determinado momento y estado, toma decisiones a partir de señales provenientes del entorno (recompensas), indicándole la medida en la que se acerca o se aleja del objetivo deseado [5]. En sencillas palabras, la política le da al robot la idea de cómo de buena es una acción en un determinado estado. Para evaluar la política se hace uso de la matriz de recompensas *Q*, la cual determina "la calidad" de una acción *a* dado un estado *s*. En el estado siguiente *s'* obtenido desde *s* al aplicar *a*, la recompensa de tomar *a* en *s* se actualiza utilizando la siguiente expresión:

$$Q(s,a) := Q(s,a) + \alpha[r + \gamma \cdot \text{argmax}_{a'}\, Q(s',a') - Q(s,a)]. \quad (1)$$

Se expresa como la suma del valor actual de *Q(s,a)*, más el producto de α (factor que indica el grado de determinismo) y la diferencia temporal (correspondiente a la diferencia entre dos estimaciones sucesivas de *Q(s,a)* de una pareja de estado-acción). Esta diferencia temporal engloba la recompensa *r* recibida debida a la acción *a* en el estado *s* (en este caso, toma valores diferentes dependiendo de si es un estado libre (r_{trans}), un destino final (r_{goal}) o con obstáculo ($r_{obj} = - r_{goal}$)), más el producto del factor de descuento γ (factor que marca el peso de la recompensa futura) y $\text{argmax}_{a'}$ *Q(s',a')* (valor máximo de la matriz *Q* en *s'*) restando *Q(s,a)*. De esta forma, trata de buscar la consecución de acciones que maximicen la recompensa (es decir, hacia aquella que le permite alcanzar el objetivo deseado) [6].

Para el cálculo del conjunto de los valores de *Q* es necesaria la adquisición de experiencia en el entorno, con el fin de poder encontrar los valores que le acerquen a su destino de forma efectiva. Este proyecto se basa en un algoritmo de aprendizaje por refuerzo que se compone de:

- **Aprendizaje por refuerzo directo (AR directo)**: se basa en el proceso de interacción con el entorno real, por el cual obtiene una serie de recompensas que le permite actualizar el modelo interno y mejorar la política mediante la actualización de la matriz *Q*.
- **Aprendizaje por planificación (AR indirecto)**: es el proceso computacional que toma el modelo interno para generar una respuesta en forma de recompensa ante una acción simulada, *ap*, en un estado simulado s_p, produciendo o mejorando la política. Es el paso fundamental para el cálculo de una matriz *Q* que nos permita calcular una trayectoria correcta.

Dyna-Q combina estos 2 métodos para alcanzar una política óptima de manera efectiva y rápida. Con la planificación obtiene la experiencia de forma simulada para encontrar los valores de recompensas más cercanas a los reales, consiguiendo una mejor política con menor interacción con el entorno. El problema de la planificación es que se basa únicamente en el modelo interno, y a veces este modelo no refleja exactamente la realidad, pudiéndose encontrar por ejemplo con obstáculos que le impidan tomar la acción más adecuada. Es aquí donde recae el uso del AR directo, ya que interactuando con el medio, actualiza o corrige el modelo permitiendo simular teniendo en cuenta todas las características reales del entorno (Figura 3).

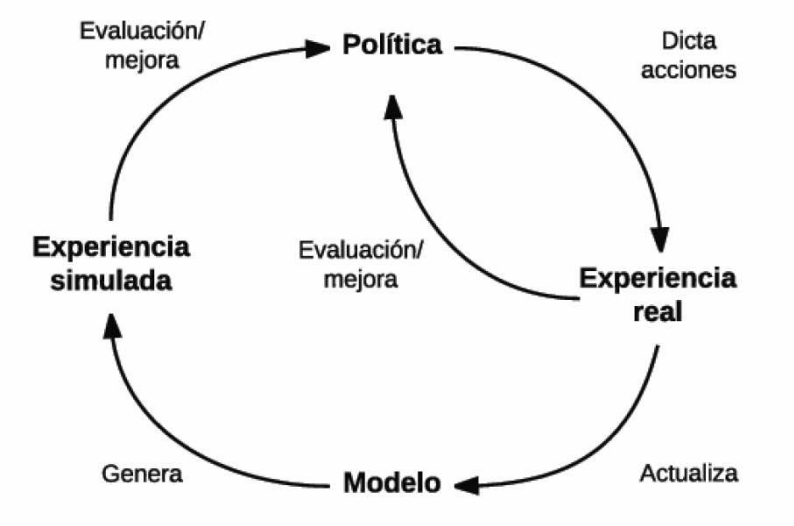


Figura 3. Arquitectura Dyna-Q [5].

3.2. Parámetros Dyna-Q

Conceptualmente, la planificación y el AR directo ocurren simultáneamente; sin embargo, su implementación en sistemas informáticos hace que estos trabajen en serie. En Dyna-Q, la acción, el AR directo y la actualización del modelo requieren poca computación, consumiendo una pequeña fracción de tiempo comparado con la

planificación, que es un proceso más intensivo. En el Algoritmo 1, se muestra la implementación en pseudocódigo de Dyna-Q para el caso de un solo robot [4].

Algoritmo 1: Dyna-Q para el sistema con un solo robot.

Entrada: Modelo interno inicial M_0, posición inicial s_0

Salida: Modelo interno M, política $\pi(s)$

1: $s := s_0$

2: **Mientras** $s \notin F$

3: $a := \pi(s)$

4: $s' := f(s,a)$ (sin ejecutar a)

5: $r := \begin{cases} r_{obj} & \text{si } s' \text{es un obstáculo} \\ r_{goal} & si\ \ s' \in F \\ r_{trans} & \text{en cualquier otro caso.} \end{cases}$

6: $Q(s,a) := Q(s,a) + \alpha[r + \gamma \cdot \text{argmax}_{a'} Q(s',a') - Q(s,a)]$

7: **Si** s' es un obstáculo

8: $O := O \cup \{s'\}$

9: Elimina transiciones a s' (actualiza f)

10: **En caso contrario**

11: **Si** s' no contiene otro robot

12: Ejecuta a

13: $s := s'$

14: **Fin si**

15: **Fin Si**

16: $s_p := s$

17: **Repetir** P veces:

18: $a_p :=$ Acción aleatoria en s_p

19: $s_p' := f(s_p,a_p)$ (Ejecuta a_p)

20: $r := \begin{cases} r_{obj} & \text{si } s_p' \text{ es un obstáculo} \\ r_{goal} & \text{si } \ s_p' \in F \\ r_{trans} & \text{en cualquier otro caso.} \end{cases}$

21: $Q(s_p, a_p) := Q(s_p, a_p) + \alpha[r + \gamma \cdot \text{argmax}_{a_p'} Q(s_p',a_p') - Q(s_p,a_p)]$

22: $s_p := s_p'$

23: **Fin Repetir**

24: **Fin mientras**

Conocido el estado inicial del robot (línea 1), se ejecuta el AR directo, donde en el estado actual s se elige una acción a del conjunto de acciones, A, siguiendo una política, $\pi(s)$, (línea 3). Se observa el estado siguiente, s', (línea 4) y se asigna una recompensa, r, que se atribuye en función de lo observado en s' (línea 5) para con ello poder actualizar Q (línea 6). Se actualiza el modelo interno (líneas 7-15) (es decir, quitando los estados del modelo ocupados por un obstáculo en caso de encontrarlo y ejecutar a en caso contrario, asignando el estado siguiente al actual) para pasar a la

fase de planificación o del AR indirecto (líneas 16-23) con el fin de generar experiencia simulada, repitiéndose un número de pasos *P*. El proceso de planificación es similar al del AR directo, pero con dos diferencias: primero la toma de acciones, a_p, es aleatoria (línea 18) y no se rige por una política (esto es debido a que el proceso de planificación trata de explorar los diferentes estados para actualizar los valores de *Q*) segundo, el estado siguiente se actualiza utilizando el modelo interno (línea 19). Una vez realizada la planificación, se repite el proceso hasta alcanzar el destino (el objetivo del robot). Todo lo ejecutado hasta llegar al destino se denomina episodio.

En cada uno de los pasos, hay distintos parámetros a definir. A continuación, se explica cada parámetro que se utiliza para entender su posible influencia en la convergencia en la búsqueda de la solución.

3.2.1. Exploración

En el conjunto de parámetros que engloba la línea 3 del Algoritmo 1, referente a la política, están los correspondientes al comportamiento del robot. Se utiliza la **política *ε*-greedy**, un método efectivo para seleccionar la acción; dada una matriz *Q*, la acción que le permite llegar a su objetivo, es decir, la que maximiza el valor *Q(s,a)*, es seleccionada con una probabilidad (1- ε), siendo entonces ε la probabilidad de que tome una acción aleatoria. A nivel de implementación se resume como:

$$\pi(s) = \begin{cases} acción\ aleatoria\ a \in A(s) & si \quad \zeta < \varepsilon \\ \operatorname{argmax}_{a \in A(s)} Q(s,a) & en\ caso\ contrario. \end{cases} \qquad (2)$$

El parámetro ζ es un valor aleatorio entre 0 y 1. Se toma una estrategia de disminuir a lo largo del tiempo el valor de ε para favorecer en los primeros instantes la exploración y con ello el encuentro de un modelo interno más exacto. Una vez obtenida cierta experiencia, se tiende hacia la elección que maximice la recompensa [7]. La forma en la que ε irá disminuyendo, será calculándolo en base a un parámetro m e inversamente proporcional al número de visitas a cada estado (3).

$$\varepsilon(s) = \frac{\mathrm{m}}{\text{número de visitas estado actual}(s)}. \qquad (3)$$

Aumentando el valor del parámetro m, ε también aumenta dando lugar a una mayor probabilidad de elegir la opción de explorar [5].

3.2.2. Cálculo

En esta sección se tienen los parámetros correspondientes a la actualización del valor *Q(s,a)* del algoritmo (Algoritmo 1, línea 6). Empezando por el principio de la función, está el **ratio de aprendizaje** α, el cual describe el grado de determinismo del entorno, que expresa cuánto peso va a tener la nueva información.

En un entorno de trabajo de AR, un robot aprende en base a las recompensas que adquiere. Consecuentemente, una secuencia de acciones que le conduce a un comportamiento no deseado produce una serie de recompensas negativas, y una secuencia que le dirige a una correcta política, recompensas positivas. Esto lleva al robot a buscar la máxima recompensa que le permita alcanzar su objetivo, por lo tanto, el valor de este **sistema de recompensas** es un parámetro a elegir/determinar; se impone este sistema en base a alcanzar el destino final en un menor número de movimientos posibles. La recompensa máxima r_{goal}, se alcanza cuando se llegue al objetivo, y por el contrario, acceder a un estado con obstáculo tendrá una recompensa $r_{obj} = - r_{goal.}$ Para realizar un menor número de movimientos a un estado que no es el objetivo, se le da una recompensa $-r_{trans}$, tal que $|r_{trans}| < |r_{obj}|$.

El parámetro γ como bien se ha definido a través de (1), es el **factor de descuento**, que determina el valor presente de las recompensas futuras con un valor de $0<\gamma<1$. Si γ es 0 solo considera las recompensas más inmediatas, sin embargo con valores más elevados de γ, las recompensas de los estados futuros tienen más importancia a la hora de actualizar *Q(s,a)*.

3.2.3. Planificación

Para terminar se tiene el bloque de *planificación*, donde en el Algoritmo 1 en la línea 17, se encuentra el parámetro *P*. Es el **número de movimientos que el robot realiza en la experiencia simulada** por cada paso que realice en el entorno real. Este parámetro es crítico, ya que influye directamente en el proceso de obtención de valores *Q(s,a)*, permitiendo obtener actualizaciones de *Q* a medida que realiza la experiencia simulada. Un número pequeño de *P*, conlleva a una menor planificación y por lo tanto a una menor actualización de los valores de *Q*, pudiendo no obtenerse una matriz de recompensas que permita obtener la trayectoria adecuada. Este valor por lo tanto tiende a ser elevado para visitar el mayor número de estados posibles en las simulaciones.

4. Planificación de trayectorias en sistemas multi-robot

En esta sección, se va a extender el algoritmo Dyna-Q que se ha desarrollado para el caso mono-robot anterior, al caso multi-robot, así como explicar las diferencias entre ambos casos. El objetivo de la misma, es el tratamiento de información para múltiples destinos y el desarrollo de un sistema de comunicación entre robots, con el fin de obtener una solución de manera más efectiva.

El funcionamiento del algoritmo en el caso multi-robot (Algoritmo 2) es similar al caso mono-robot (Algoritmo 1), pero plantea 2 diferencias:

- Si hay varios destinos alcanzables s_d, la matriz Q de cada robot tiene que almacenar las recompensas calculadas en base a cada destino. La matriz asociada a cada destino se denomina Q^d.
- Si hay varios robots, es conveniente que cooperen entre ellos en lugar de ejecutar individualmente el algoritmo Dyna-Q (interacción apática), con el fin de ayudarse mutuamente a la consecución del objetivo (alcanzar los destinos). Por lo que en este caso se incluye la comunicación entre robots.

La primera diferencia con el caso mono-robot se puede ver en las líneas 6-9 y 23-26, donde en AR directo y planificación se evalúan las recompensas para cada destino. El objetivo del robot es alcanzar un destino, por lo tanto tiene que evaluar cada uno de ellos, y esto lo hace con ayuda de la matriz Q^d de cada destino. La segunda diferencia, se puede observar en la línea 3, donde se ha colocado la comunicación entre robots, que se produce a través de un protocolo de comunicación. Se ha colocado antes de la política $\pi(s)$ (línea 4) con el fin de tener en cuenta, antes de elegir una acción, la información aportada por los demás robots.

La comunicación consiste en el intercambio de información distinta entre dos robots. Algunos ejemplos de información que se pueden intercambiar son: los obstáculos del mapa, los destinos finales alcanzados y la matriz de recompensas. Tanto los obstáculos como los destinos finales serán tratados en todos los protocolos desarrollados por igual, pero la matriz de recompensas será analizada desde diferentes puntos de vista.

Algoritmo 2: Dyna-Q en un sistema multi-robot ejecutado por el robot *i.*

Entrada: Modelo interno inicial M_i^0, posición inicial s_0

Salida: Modelo interno M_i, política $\pi(s)$

1: $s := s_0$

2: **Mientras** $s \notin F_i$

3: Intercambio de información **Si** distancia entre robots<R

4: $a := \pi(s)$

5: $s' := f_i(s,a)$ (sin ejecutar a)

6: **Para** cada $s_d \in F_i$ no alcanzado

7: $$r := \begin{cases} r_{obj} & \text{si } s' \text{es un obstáculo} \\ r_{goal} & si\ \ s' = s_d \\ r_{trans} & \text{en cualquier otro caso.} \end{cases}$$

8: $Q_i^d(s,a) := Q_i^d(s,a) + \alpha[r + \gamma \cdot \text{argmax}_{a'} Q_i^d(s',a') - Q_i^d(s,a)]$

9: **Fin Para**

10: **Si** s' es un obstáculo

11: $O_i := O_i \cup \{s'\}$

12: Elimina transiciones a s' (actualiza f_i)

13: **En caso contrario,**

14: **Si** s' no contiene otro robot

15: Ejecuta a

16: $s := s'$

17: **Fin Si**

18: **Fin Si**

19: $s_p := s$

20: **Repetir** P veces:

21: $a_p :=$ Acción aleatoria en s_p

22: $s_p' := f_i(s_p,a_p)$ (Ejecuta a_p)

23: **Para** cada $s_d \in F$ no alcanzado

24: $$r := \begin{cases} r_{obj} & \text{si } s_p' \text{es un obstáculo} \\ r_{goal} & si\ \ s_p' \in F \\ r_{trans} & \text{en cualquier otro caso.} \end{cases}$$

25: $Q_i^d(s_p,a_p) := Q_i^d(s_p,a_p) + \alpha[r + \gamma \cdot \text{argmax}_{a_p'} Q_i^d(s_p',a_p') - Q_i^d(s_p,a_p)]$

26: **Fin Para**

27: $s_p := s_p'$

28: **Fin Repetir**

29: **Fin Mientras**

4.1. Gestión de los múltiples destinos

La toma de decisiones de los robots se basa en la matriz de recompensas *Q*, eligiendo las acciones que le permitan maximizar este valor, dirigiéndolo al objetivo.

Como existen varios objetivos alcanzables, cada robot *i* lleva consigo una Q^d asociada a cada destino. Cada robot tiene que evaluar la política para varios destinos, eligiendo la acción que maximice la recompensa sobre todos los destinos. Esto se ha llevado a cabo calculando, dado un estado *s*, el sumatorio de $Q^d(s,a)$ de todos los destinos para cada una de las acciones $a \in A$, obteniendo una $Q_{total}(s,a)$ que se utiliza para evaluar la política. El valor de los sumatorios en cada *a* refleja los efectos de todas las acciones para todos los destinos (Algoritmo 3), lo que permite dirigir al robot hacia el destino más cercano.

Algoritmo 3: Tratamiento de *Q* para robot *i*.

1: **Para** cada $a \in A$ en *s*

2: $Q_{total}(s,a) := \sum_d Q^d(s,a) \ \forall \quad d$=1,2,…,*N* no alcanzado

3: **Fin para**

Detectado un robot *j* dentro del radio de comunicación de un robot *i*, se intercambian la información de los destinos alcanzados por ellos mismos o por los distintos robots. Es decir, cada uno de ellos en el conjunto *F*, además de la posición de los destinos, poseen la información relativa a si han sido alcanzados. A medida que exploran y se comunican, averiguan los destinos que han sido ocupados. Esto permite al robot orientar su política hacia otros destinos y evitar los cálculos relativos a la actualización de la matriz Q^d de un destino alcanzado, reduciendo el tiempo de ejecución.

4.2. Actualización del mapa de obstáculos

En los primeros instantes, el desconocimiento de los obstáculos es total y sus acciones tienden a la exploración. Se debe a que el parámetro ε tal y como se calcula en (3), tiende a ser elevado en los primeros movimientos, por lo que si se observa (2), hay mayor probabilidad de que tome una acción aleatoria, es decir, que explore. A medida que aumenta el número de visitas a cada estado, es más probable que dirija sus acciones hacia un destino, pudiendo no tener la información completa del mapa.

Como el entorno es estático, la información obtenida mediante exploración por cada robot puede ser explotada de manera sencilla por el resto de miembros del equipo. Esto permite tener a cada robot una información del mapa más rápida y completa a medida que se comunican entre los diferentes robots. Este primer protocolo desarrollado está basado en la transmisión del conjunto de obstáculos *O* de los robots, que almacena la posición de estos en el entorno.

Si la distancia entre dos robots *i* y *j* es menor al alcance del radio de comunicación *R*, se intercambian el conjunto *O* y *F* que cada uno de ellos tiene almacenado en su modelo interno (Protocolo 1).

Protocolo 1: (ComObs) [5] ejecutado por robot *i.*	
1:	Recibir O_j y F_j
2:	$O_i := O_i \cup O_j$ <<actualiza el conjunto de obstáculos>>
3:	$F_i := F_i \cup F_j$ <<actualiza el conjunto de destinos finales alcanzados>>
4:	Actualizar f_i

Esto permite a los robots tener un modelo interno más completo, mejorando el proceso de planificación, lo que conlleva a evitar pasos hacia estados ocupados por obstáculos y con ello hacia una mejora más rápida de la política.

4.3. Actualización de las matrices de recompensa

A la hora de actualizar la matriz de recompensas, se hace uso del consenso entre robots para determinar los nuevos valores que permitan de distintas formas mejorar el resultado de la matriz. La matriz de recompensas se basa en cálculos que, bajo el punto de vista de distintos robots puede tener un resultado distinto, por lo que hay que formular un consenso para tratar de la manera más adecuada esta información. Esto no pasa con los obstáculos y destinos finales, ya que al tratarse de un entorno determinista, su existencia es igual para todos los robots.

4.3.1. Comunicación con la media aritmética

El consenso entre los robots se realiza a través de una *media aritmética* de la matriz de recompensas que tengan guardada actualmente (4), lo que da paso al Protocolo 2. Se considera por lo tanto que la información conseguida por cada robot es igual de relevante, es decir, tiene el mismo peso a la hora de realizar el consenso.

$$Q_i^d(s,a) := \frac{Q_i^d(s,a) + Q_j^d(s,a)}{2}. \qquad (4)$$

Protocolo 2: (ComMed) ejecutado por robot *i.*	
1:	Recibir O_j, F_j y Q_j
2:	$O_i := O_i \cup O_j$ <<actualiza el conjunto de obstáculos>>
3:	$F_i := F_i \cup F_j$ <<actualiza el conjunto de destinos finales alcanzados>>
4:	Actualizar f_i
5:	Ejecutar (4), $\forall$ d=1,2,…,N

Como aspecto positivo se puede decir que toda la información conseguida por cada robot tiene la misma importancia sin desechar o despreciar información. Esta comunicación de la matriz de recompensas puede ayudar a encontrar un estado final desocupado a un robot *i* gracias a la experiencia adquirida por otro robot *j* en la búsqueda de la trayectoria hacia cierto destino.

[5] Los nombres dados a los distintos protocolos sirven para ayudar al posterior análisis en las gráficas.

4.3.2. Comunicación con la media ponderada

En el siguiente protocolo, la información de *Q* de dos robots *i* y *j* en comunicación ya no tienen el mismo peso. Dado un estado *s* y una determinada acción *a* que conduce al mismo estado siguiente *s'*, se puede asumir que el robot que más veces haya realizado dicha acción *a* tendrá una mayor experiencia. Esto lo podemos considerar en el protocolo de consenso calculando una *media ponderada* de las matrices (5), eligiendo pesos iguales al número de veces que cada robot ha realizado la acción evaluada.

Tanto en éste como en el siguiente protocolo, se suma el total de las acciones *a* realizadas desde *s*, *Z(s,a)*, y las visitas a cada estado, *V(s)*, (valores almacenados en la matriz *Q*) para asociarlo a ambos robots (línea 8 y 10 respectivamente). Con esto se consigue que el robot se decida a buscar antes la trayectoria hacia un destino final, según la política que se ha definido, antes que la exploración (Protocolo 3).

$$Q_i^d(s,a) := \frac{Q_i^d(s,a) * Z_i^d(s,a) + Q_j^d(s,a) * Z_j^d(s,a)}{Z_i^d(s,a) + Z_j^d(s,a)}. \qquad (5)$$

Protocolo 3: (ComPrio) ejecutado por robot *i.*

1: Recibir O_j, F_j y Q_j
2: $O_i := O_i \cup O_j$ <<actualiza el conjunto de obstáculos>>
3: $F_i := F_i \cup F_j$ <<actualiza el conjunto de destinos finales alcanzados>>
4: Actualizar f_i
5: **Para** cada $s \in S$
6: **Para** cada $a \in A$
7: Ejecutar (5) , $\forall$ d=1,2,…,N
8: $Z_i^d(s,a) := Z_i^d(s,a) + Z_j^d(s,a)$, $\forall$ d=1,2,…,N
9: **Fin Para**
10: $V_i(s) := V_i(s) + V_j(s)$
11: **Fin Para**

La suma de las acciones evita que si se dan dos comunicaciones consecutivas entre dos mismos robots, una vez calculada la nueva matriz de recompensas para ambos, no se subestimen o sobreestimen los nuevos valores adquiridos en *Q* de uno de los robots. Sumar además las acciones realizadas y las visitas es lógico, ya que entre ambos una vez realizada la comunicación se obtiene una experiencia conjunta. El uso de la media ponderada además, ayuda a los robots que reciben la información a la obtención de valores más precisos hacia destinos donde han tenido una menor experiencia.

4.3.3. Comunicación condicional

Este protocolo ya no realiza un cálculo con la matriz de recompensas como los dos anteriores, si no que se hace una elección de la información que considera más adecuada. En concreto, los robots se comunican la matriz de recompensas y se toma como valor para ambos el del robot que mayor número de acciones *a* haya tomado hacia el estado siguiente *s'* (6). Este caso desecha la información del robot menos experimentado realizando dicha acción, importando solo la información del robot con mayor experiencia (Protocolo 4).

$$\begin{aligned} &Si\ Z_j^d(s,a) > Z_i^d(s,a) && Q_i^d(s,a) := Q_j^d(s,a) \\ &En\ caso\ contrario && Q_j^d(s,a) := Q_i^d(s,a). \end{aligned} \qquad (6)$$

Protocolo 4: (ComCon) ejecutado por robot *i.*

1: Recibir O_j, F_j y Q_j
2: $O_i := O_i \cup O_j$ <<actualiza el conjunto de obstáculos>>
3: $F_i := F_i \cup F_j$ <<actualiza el conjunto de destinos finales alcanzados>>
4: Actualizar f_i
5: **Para** cada $s \in S$
6: **Para** cada $a \in A$
7: **Si** $Z_j^d(s,a)$ es mayor a $Z_i^d(s,a)$
8: $Q_i^d(s,a) := Q_j^d(s,a)$, ∀ *d*=1,2,…,*N*
9: $Z_i^d(s,a) := Z_i^d(s,a) + Z_j^d(s,a)$, ∀ *d*=1,2,…,*N*
10: **Fin si**
11: **Fin Para**
12: $V_i(s) := V_i(s) + V_j(s)$
13: **Fin Para**

Este protocolo puede ser interesante cuando las comunicaciones se producen en menor cantidad, debido a que además de solo considerar la información más relevante, la cantidad de información desechada no sería muy elevada. Con el objetivo de reducir la cantidad de información que se desecha y comprobar si ayuda a la obtención de una solución, también se analizará una versión del protocolo donde se decide cuando se comunican (Protocolo 4.2, línea 7). Si la diferencia en el número de acciones *a* de dos robots *i* y *j* a un mismo estado siguiente *s'* es mayor a cierto número λ, se procede al intercambio de la información de su matriz de recompensas.

Protocolo 4.2: (ComCon2) ejecutado por robot *i.*

1: Recibir O_j, F_j y Q_j
2: $O_i := O_i \cup O_j$ <<actualiza el conjunto de obstáculos>>
3: $F_i := F_i \cup F_j$ <<actualiza el conjunto de destinos finales alcanzados>>
4: Actualizar f_i
5: **Para** cada $s \in S$
6: **Para** cada $a \in A$
7: **Si** $Z_j^d(s,a)$ es mayor a $Z_i^d(s,a)$ y $|Z_i^d(s,a) - Z_j^d(s,a)|$ mayor a λ
8: $Q_i^d(s,a) := Q_j^d(s,a)$, $\forall$ d=1,2,…,N
9: $Z_i^d(s,a) := Z_i^d(s,a) + Z_j^d(s,a)$, $\forall$ d=1,2,…,N
10: **Fin si**
11: **Fin Para**
12: $V_i(s) := V_i(s) + V_j(s)$
13: **Fin Para**

5. Simulación y análisis de los protocolos

En esta sección, se explican las diferentes herramientas que se utilizan para realizar las simulaciones y analizar los datos, así como los resultados obtenidos a partir de ellos. La finalidad principal de la misma es evaluar los resultados, lo cual va a permitir extraer las principales conclusiones de este trabajo.

5.1. Entorno y herramientas de trabajo

Se ha elegido el entorno informático *Matlab*. Dentro de *Matlab*, se ha trabajado con la librería *Robotic Motion Toolbox* (RMTool)[6] [8], una herramienta interactiva de código abierto, que permite entre otras cosas la planificación de trayectorias, el control de movimiento y la modelización. Tiene una interfaz gráfica (Figura 4) para poder ser utilizado por un usuario sin experiencia previa en *Matlab*, interactuando solo con la herramienta, creando el entorno, eligiendo el tipo de mapa (descomposición rectangular, cuadrangular…), entre otras funcionalidades que no son necesarias para este trabajo.

Además, RMTool permite la adición de funciones/algoritmos, lo cual ha permitido implementar la función Dyna-Q y sus respectivas funciones para simular el modelo y la detección de obstáculos.

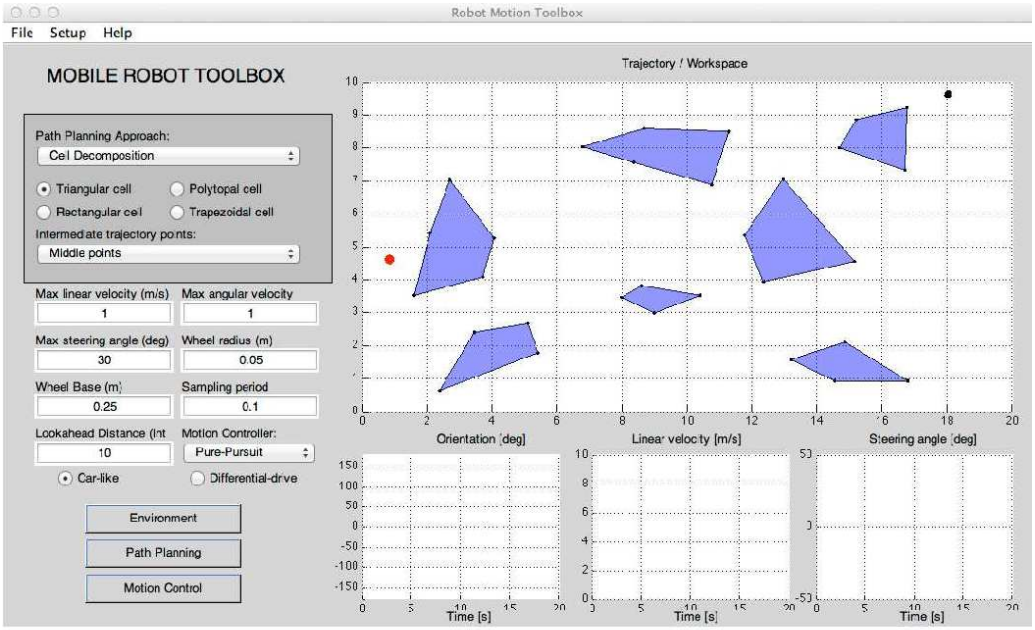


Figura 4. Entorno gráfico de trabajo RMTool [8].

Las simulaciones realizadas en este proyecto requieren un enorme tiempo de computación, donde las operaciones realizadas se basan en multitud de bucles. Esta razón ha llevado a solicitar acceso al *Clúster Hermes*, un clúster de computación de alto rendimiento del I3A (Instituto de Investigación en Ingeniería de Aragón) que utiliza

[6] http://webdiis.unizar.es/RMTool/

Condor, un sistema que ofrece funcionalidades de paradigma HTC (High Throughput Computing).

MobaXterm es el software que permite trabajar con el clúster bajo una interfaz textual Linux, con una comunicación vía SSH. Con él se lanzan las simulaciones, se comprueba el funcionamiento de los programas dentro del clúster y permite visualizar los ficheros generados. Otro programa utilizado es *WinSCP*, para el intercambio de ficheros entre el ordenador personal y el clúster, permitiendo transferir los archivos en mayor cantidad, con mayor facilidad y organizar los ficheros dentro del sistema de almacenamiento que el usuario tenga asociado en el clúster vía SFTP.

5.2. Simulaciones

Para comprobar en primera instancia la efectividad de los protocolos, se realizan una serie de simulaciones. También va a permitir estudiar la mejor alternativa en el Protocolo 4.2. Se han hecho las pruebas en 20 mapas distintos de 20x20 (Figura 5), con 10 réplicas, donde 5 robots tratan de llegar a 5 destinos finales. Previamente al movimiento de los robots, cada robot divide el entorno en un conjunto de estados, el cual se lleva acabo informando a cada uno de su tamaño y del tamaño de cada estado. Los robots discretizan el entorno en cuadrados de iguales dimensiones, limitando el movimiento a los estados adyacentes al estado actual, es decir, a los puntos cardinales (N-S-E-O).

Los obstáculos se posicionan aleatoriamente en el entorno, ocupando un determinado número de estados (dependiente del tamaño del obstáculo). La detección de obstáculos y de otros robots se realiza por medio de la simulación de un sistema de sensores implementados en los robots, permitiendo localizarlos en los estados circundantes al que se encuentre. Como en este caso los robots se manejan en un entorno determinista, el parámetro α es igual a 1. La comunicación se produce si la distancia euclídea, δ, entre dos robots es menor al radio de comunicación *R.*

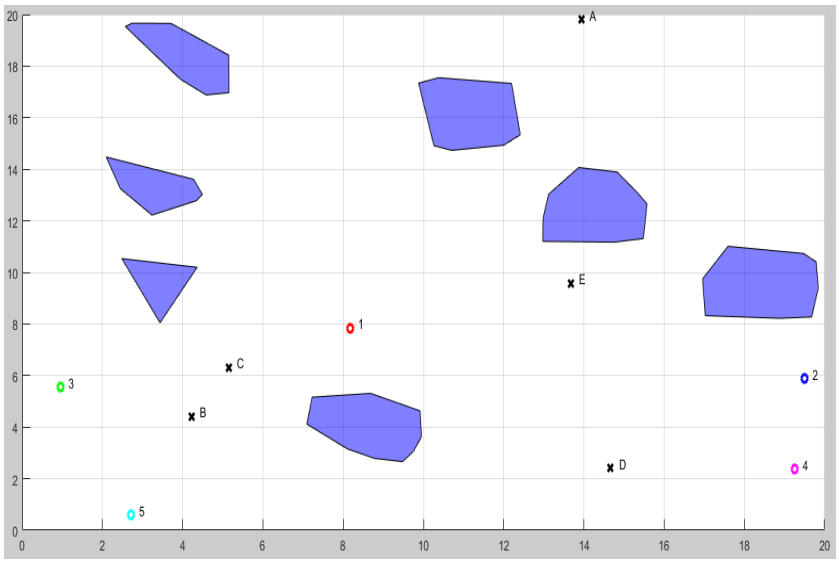


Figura 5. Ejemplo de mapa 20x20 aleatorio utilizado.

Se calcula el recorrido total de los 5 robots con cada protocolo para cada radio de comunicación, *R*. Se obtiene una media total de los mapas y replicas para cada *R* (de 1 a 10) de los robots para ver como varía la efectividad de los distintos protocolos cuando *R* aumenta. Esta efectividad se mide respecto al algoritmo Dyna-Q sin comunicación, calculando la mejora relativa (7). En casos donde se realice una comparación con mapas y número de robots de tamaño distinto, se utiliza también la mejora absoluta (8).

$$Mejora\ relativa = \frac{Media\ \ sin\ comunicación - Media\ con\ comunicación}{Media\ sin\ comunicación}. \qquad (7)$$

$$Mejora\ absoluta = Media\ \ sin\ comunicación - Media\ con\ comunicación. \qquad (8)$$

Como parámetros del algoritmo Dyna-Q (Tabla 1), se eligen valores similares a los encontrados en [5].

Parámetros	m	γ	*P*
Valor	1	0.9	5000

Tabla 1. Valores predefinidos de los parámetros para simulaciones.

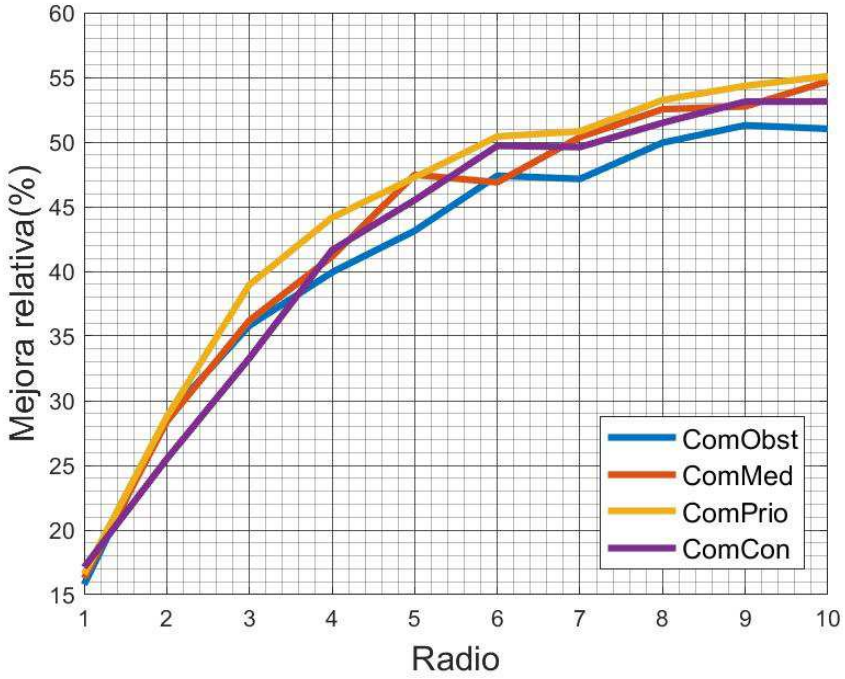
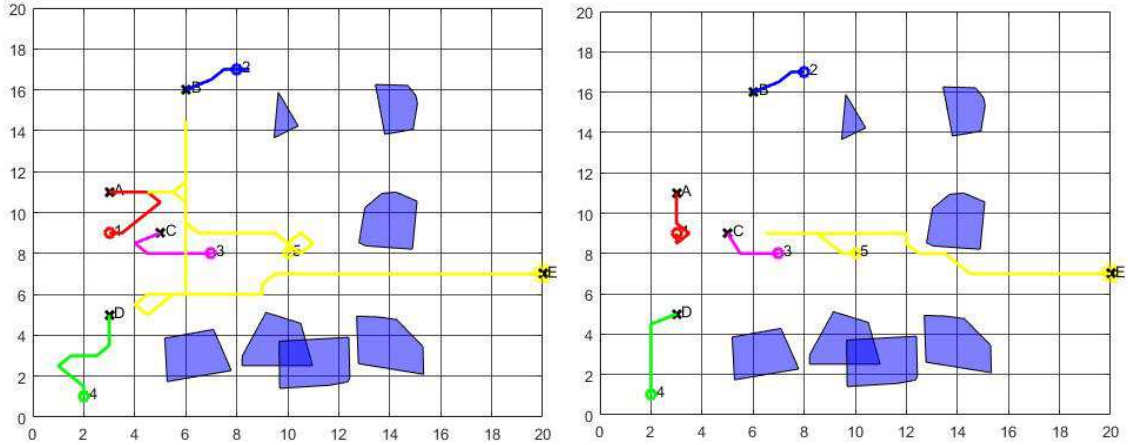


Figura 6. Mejora respecto a modelo sin comunicación.

Como se puede observar en la Figura 6, los protocolos tienden a mejorar aumentando el radio de comunicación. La mejora se debe al ahorro de pasos en falso a estados donde existen obstáculos, o destinos que ya han sido alcanzados, además de llegar a un mejor resultado a la hora de calcular la matriz de recompensas. Se debe a que realizando un consenso entre los robots logran obtener una mejor matriz *Q* que les permitan conseguir una política más adecuada, y con ello alcanzar el destino de

manera más rápida (Figura 7, donde hay una diferencia de 37 pasos entre a) y b)). Vemos que los 4 protocolos empiezan a converger a partir del radio 9, donde se alcanza prácticamente una comunicación central, ocupando el alcance de *R* casi la totalidad del mapa.



a) Trayectoria sin comunicación. b) Trayectoria con comunicación.

Figura 7. Ejemplo de trayectorias.

El Protocolo 4 (ComCon), presenta los mejores resultados cuando la comunicación es inusual (*R* =1), debido a que al comunicarse menos los robots, solo se transmiten la información relevante. A medida que se aumenta el radio sigue siendo mucho más efectivo que sin comunicación, pero en menor medida que los otros 2 protocolos que calculan la media de las matrices *Q*. Esto se debe al enorme desecho de información que se realiza, pudiendo no permitir en los primeros momentos adquirir una experiencia propia antes de guiarse por la información del resto de robots.

Por esta razón, se pretende realizar simulaciones de dicho protocolo eligiendo cuando deben comunicarse (Protocolo 4.2). Lo que se quiere confirmar es si una disminución de la cantidad de información que se desecha permite una mejora en el rendimiento, provocando que los robots tengan más tiempo para calcular sus propios valores *Q(s,a)*, obteniendo así una información más adecuada desde la perspectiva de cada robot.

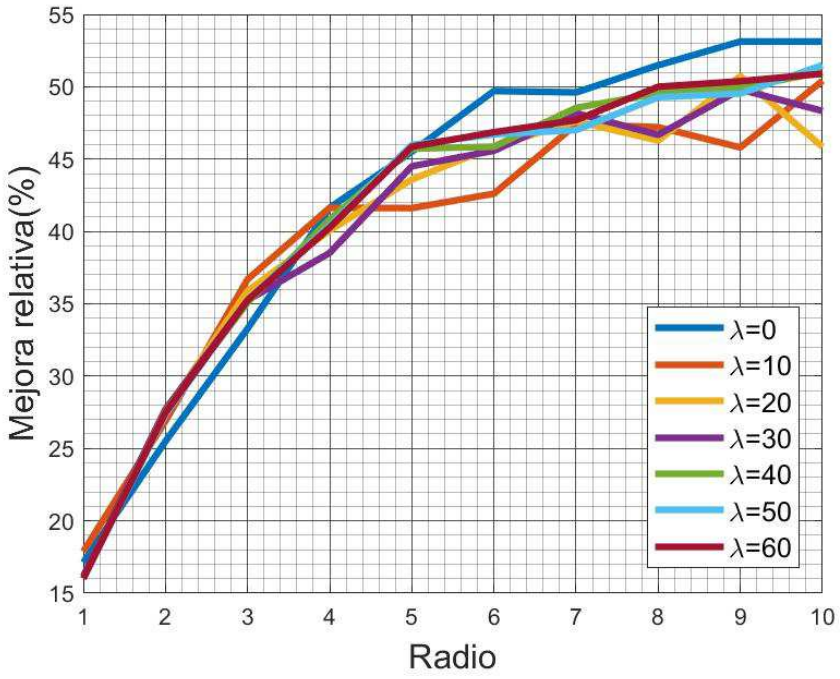


Figura 8. Eficacia de la elección del momento de comunicación.

Como puede observarse en la Figura 8, hay una mayor eficacia comunicándose siempre que se cumpla $\delta<R$, es decir, a una diferencia mayor o igual a 0 entre las *Z(s,a)* de los robots *i* y *j* (λ=0). Hay que añadir que esta opción tiene como consecuencia una enorme cantidad de transmisión de datos (Figura 9), disminuyendo cuando se aumenta esta diferencia. Con diferencia 50 y 60 (λ=50 y λ=60), se podría considerar una comunicación semejante al Protocolo 1, donde se comunican la posición de los obstáculos y destinos alcanzados, ya que no muestra una cantidad muy grande de información transmitida. Debido al carácter estocástico del problema, se observan algunas anomalías presentadas en algunas soluciones, como disminuciones en la mejora al aumentar el radio debido posiblemente a algún resultado negativo, que ha supuesto un peso significativo en la media total (Figura 8 λ=10, Radio=9), o la tendencia a disminuir la cantidad de información transmitida en los primeros radios de comunicación (Figura 9).

Como se está ante una metodología en la que únicamente nos basamos en la obtención de una mayor eficacia por parte de los protocolos, además de usar un entorno de simulación, de aquí en adelante se usa λ=0. No obstante, la cantidad de información a la hora de utilizarlo en aplicaciones reales es un parámetro a tener en cuenta, debido al coste que supondría tanto en el tiempo de ejecución como tecnológico una mayor transmisión de información.

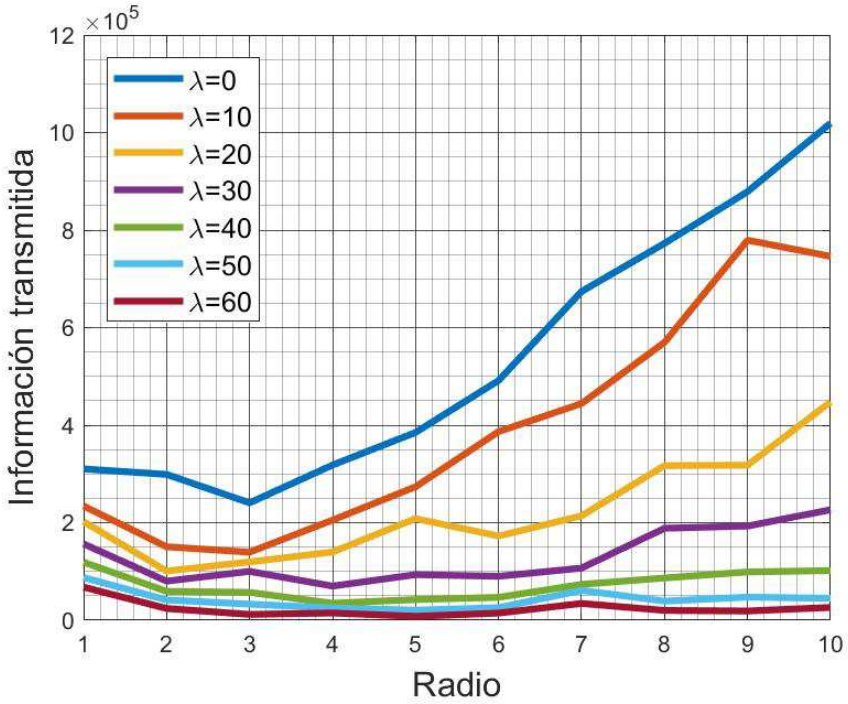


Figura 9. Ancho de banda.

5.3. Comportamiento en distintos entornos

Comprobado el correcto funcionamiento de los protocolos y la mayor eficiencia en los sistemas multi-robot funcionando con protocolos de comunicación, se analiza cómo se comportan al aumentar el tamaño del mapa (cambiando a un mapa 40x40, con mayor número de obstáculos y ligeramente más grandes, Figura 10 con 3 robots), y al aumentar el número de robots (6 robots en Figura 11) viendo como varía el rendimiento al implementar estos cambios. La elección de este número de robots y este tamaño de mapa (y no números más altos) es debido a los enormes tiempos de ejecución para realizar las simulaciones (incluyendo en el clúster). A la vez, se compara en la misma gráfica con los mapas 20x20, con la finalidad de poder apreciar la diferencia en la eficiencia de los distintos tipos de entorno y número de robots.

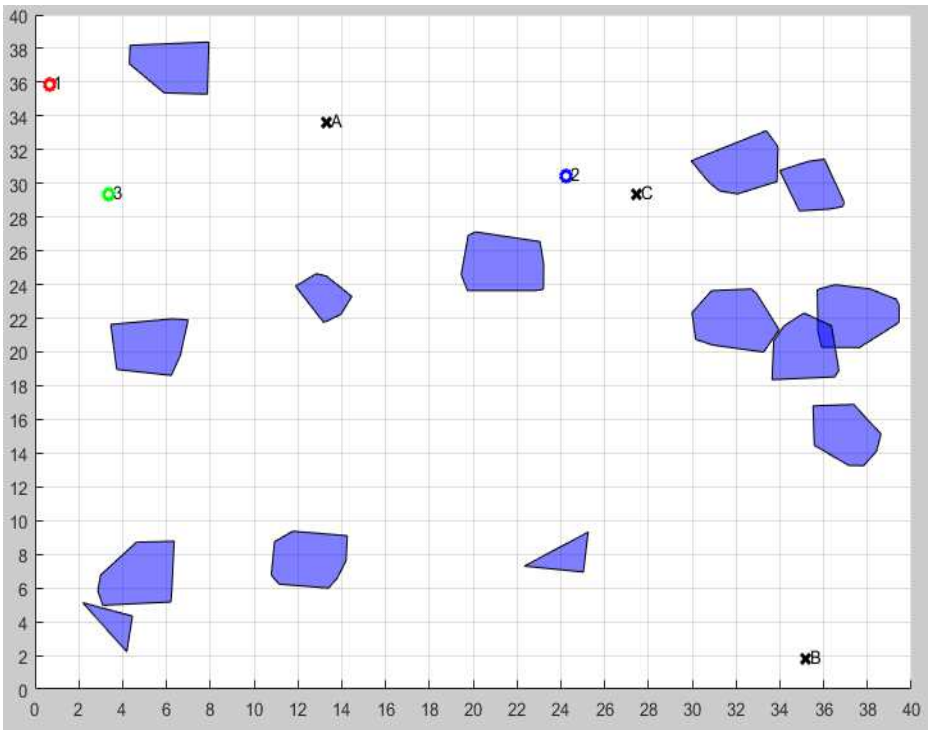


Figura 10. Mapa 40x40 con 3 robots.

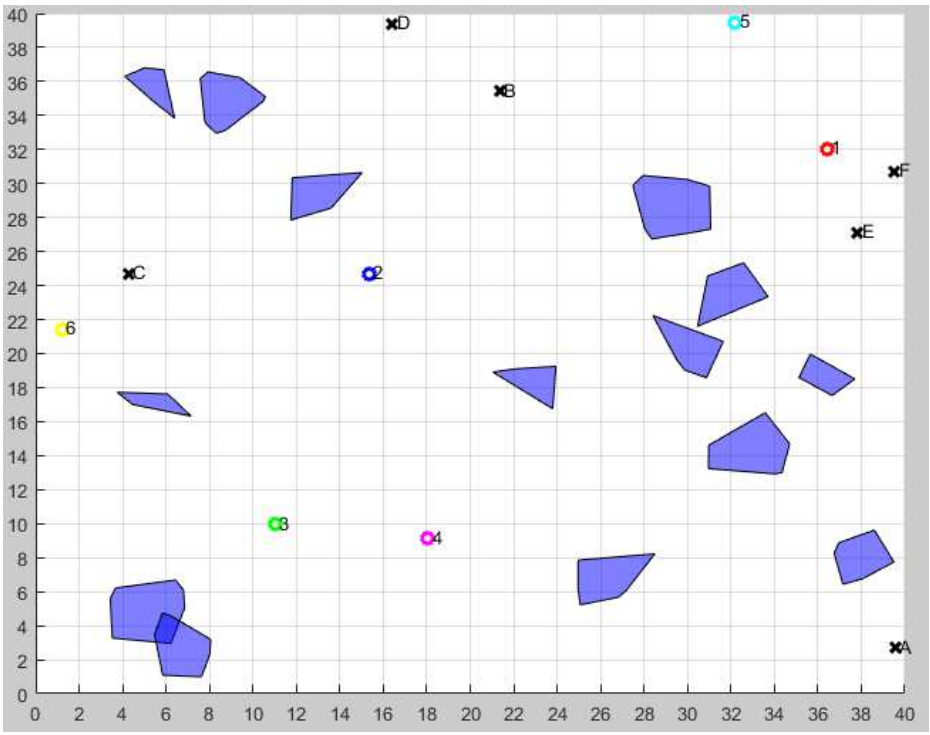


Figura 11. Mapa 40x40 con 6 robots.

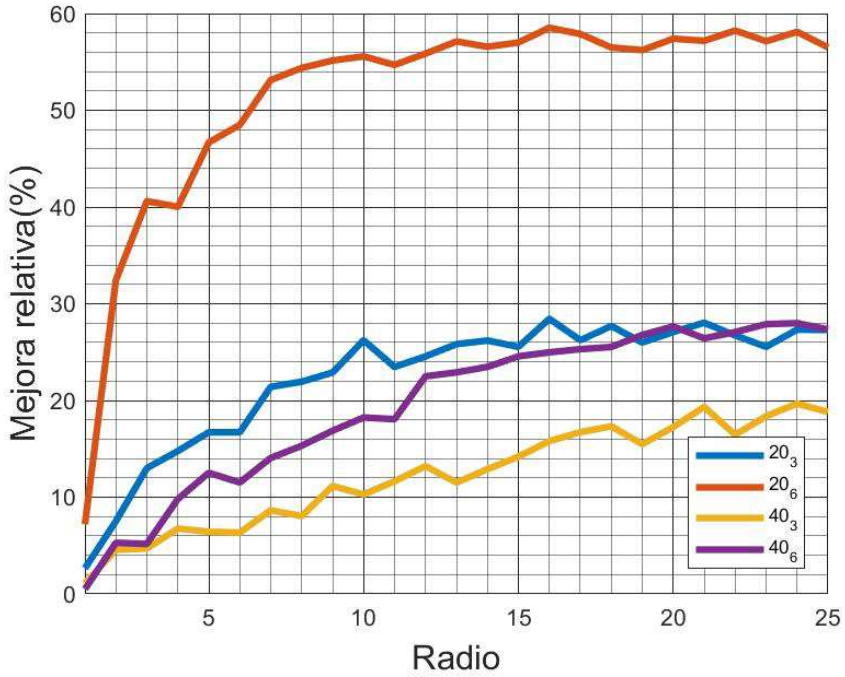


Figura 12. Mejora relativa Protocolo 3.

Se puede observar en la Figura 12 que el comportamiento es similar al visto en la gráfica de la Figura 6, tendiendo a crecer hasta alcanzar la convergencia. Para un mayor número de robots, la mejora es más notable, sobre todo en los mapas 20x20 de 6 robots. Esto es lógico, debido a que los robots están más concentrados, pudiendo ayudarse mutuamente a llegar antes a su destino a mayores velocidades. Con objeto de ayudar más a comparar los distintos entornos y número de robots, se ha representado también en términos absolutos (8), ya que en términos relativos la diferencia de pasos en 20x20 tiene mayor relevancia en términos relativos que en 40x40 (Figura 13).

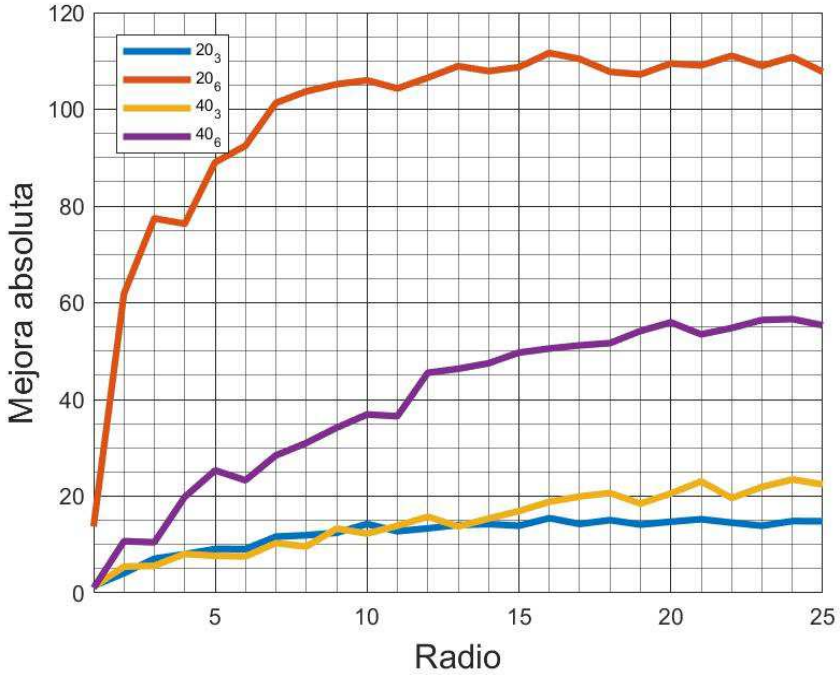


Figura 13. Mejora absoluta Protocolo 3.

En los apéndices A y B están las gráficas correspondientes a los demás protocolos, los cuales a excepción del Protocolo 4 (Figura 14) tienden a tener un comportamiento similar al del Protocolo 3 (ComPrio) con un menor rendimiento. Este protocolo presenta un problema a la hora de llevarlo a los mapas 40x40 y 6 robots, viendo que a partir del radio 15 empieza a reducirse la mejora, tendiendo a empeorar la búsqueda de la trayectoria. Analizando los distintos resultados en distintos mapas, se pudo observar que había pruebas donde el número de pasos dados en total por los robots era muy grande, influyendo así en la media total de las simulaciones. Al aumentar el número de robots y el mapa, los robots pueden entorpecerse entre sí haciendo que este robot cambie de destino, aumentando así el número de pasos realizado y con ello bajando la eficiencia. Este comportamiento se puede ver también en el resto de mapas y número de robots, viendo que no tiende a estar en un valor continuo sino a tener un número elevado de picos pronunciados.

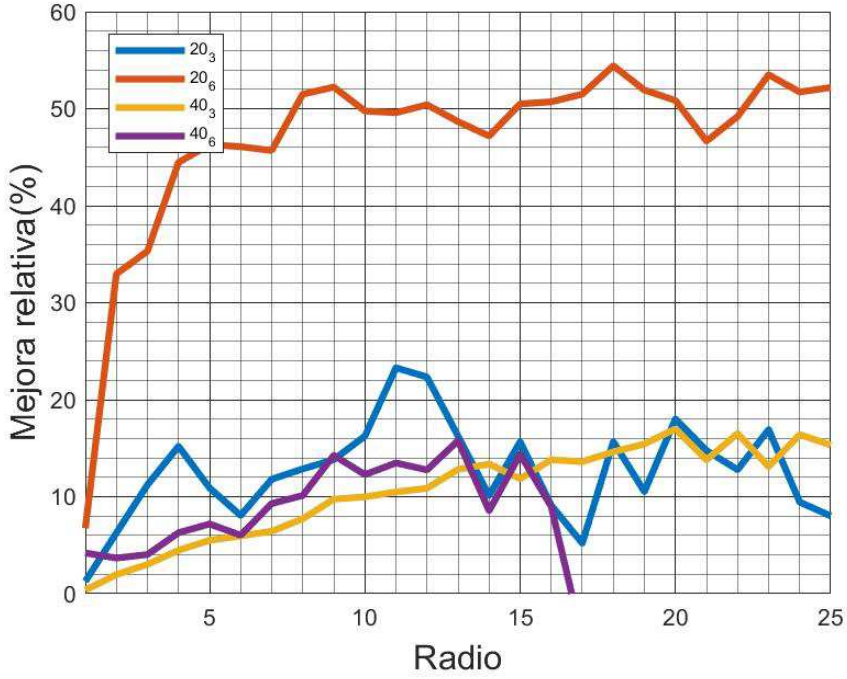


Figura 14. Mejora relativa Protocolo 4.

6. Conclusiones

En este trabajo se ha estudiado el problema de planificación de trayectorias en un sistema multi-robot en entornos desconocidos, utilizando un algoritmo de aprendizaje por refuerzo junto con métodos de consenso distribuido.

A nivel de los objetivos planteados inicialmente en la propuesta se puede determinar qué:

- Se ha conseguido extender el algoritmo Dyna-Q planteado en un sistema multi-robot considerando la cooperación entre los robots. Se han propuesto protocolos de comunicación para mejorar la política durante la ejecución del algoritmo de aprendizaje, permitiendo reducir el número de pasos realizados por el conjunto de robots.
- Se han analizado los distintos protocolos de comunicación diseñados para demostrar la mejora en el comportamiento de los robots dependiendo de la información que se transmiten, observando también la enorme cantidad de información que se comparte.

De cara a los resultados obtenidos en la simulación de Dyna-Q con los protocolos de comunicación implementados, se pueden extraer las siguientes conclusiones:

- Realizando un sistema de cooperación entre robots intercambiando únicamente la información de obstáculos y posiciones finales ocupadas (Protocolo 1), trae consigo una mejora considerable en el rendimiento.
- Un mayor radio de comunicación orienta a una mejora en el rendimiento, debido a la mayor probabilidad de comunicarse antes la información. Un radio muy grande, conlleva una transmisión de gran cantidad de información en los protocolos 2, 3 y 4 (donde se intercambian las matrices *Q)*. En una aplicación real, esto debe tenerse en cuenta debido al coste tecnológico que supone, tanto la cantidad de datos transmitidos como el alcance del radio de comunicación.
- Los métodos donde se calcula la media de las matrices de recompensa (Protocolo 2 y 3) han demostrado en estas pruebas tener mayor rendimiento que el método donde se elige los valores de *Q* más adecuados (Protocolo 4). Este último es útil donde tan solo se comuniquen en estados adyacentes, debido a la poca información que se desecha en estos rangos.
- Al aumentar el número de robots y la concentración de ellos en un entorno, mejora notablemente el rendimiento respecto a un sistema multi-robot sin comunicación.
- El Protocolo 1 de comunicación de obstáculos ha demostrado ser eficaz en pequeños mapas, con un ancho de banda mucho más reducido al de los otros protocolos. A medida que el tamaño de mapa aumenta, se aprecia con más diferencia un mayor rendimiento de los protocolos 2 y 3.

Durante las simulaciones, se han observado algunos comportamientos extraños. Por ejemplo, una simulación con una política no adecuada o una comunicación ineficiente produce trayectorias que perjudican la media total, viendo un resultado inesperado. Aun con este inconveniente, la utilización de la media y la mejora relativa han sido efectivas para interpretar los resultados. El empleo de la mejora absoluta en entornos y número de robots distintos ha sido adecuado para analizar el rendimiento, permitiendo comparar los resultados de manera más eficaz.

El entorno *Matlab* y la herramienta RMTool, junto con la implementación de Dyna-Q ha resultado ser apropiada para un número de robots no muy elevado, ya que el tiempo de simulación a medida que aumenta el número de robots crece de manera pronunciada.

Aunque las modificaciones realizadas han resultado efectivas, cabe decir que quedan incógnitas por resolver, como puede ser la influencia de los parámetros del algoritmo Dyna-Q, que se han tomado de [5] en base a los resultados obtenidos. Estos parámetros estaban calculados para un determinado entorno bajo un cierto número de robots distinto a los utilizados aquí, por lo que puede encontrarse una mejora aun más alta de eficiencia encontrando los nuevos parámetros de Dyna-Q. Debido al alto tiempo de simulación que conllevaría para cada número de robots y entornos distintos, siendo mi principal objetivo el estudio de los protocolos de comunicación y los sistemas multi-robot en cooperación, se ha preferido centrar los esfuerzos en esta línea y dejar este apartado a futuros estudios.

Bibliografía

[1] Zimmer, M. (2018). *Apprentissage par renforcement développemental.* Tesis Doctoral. Université de Lorraine.
Disponible en : https://tel.archives-ouvertes.fr/tel-01735202/document

[2] Heras-Godínez,J., Velasco Villa, M., Vásquez, J.A. (2014). "Problema de Consenso en Redes de Robots de Primer Orden con Retardo en la Comunicación". *Memorias del XVI Congreso Latinoamericano de Control Automático*, Cancún, Quintana Roo, México 14-17 Octubre 2014. Veracruz: Universidad Veracruzana, pp. 272-277.
Disponible en: http://amca.mx/memorias/amca2014/media/files/0186.pdf

[3] Guoguang Wen (2012). *Distributed cooperative control for multi-agent systems*. Tesis Doctoral. Ecole Centrale de Lille.
Disponible en: https://tel.archives-ouvertes.fr/tel-00818774/document

[4] Sutton, R.S. y Barto, A.G. *Reinforcement Learning: An Introduction.* The MIT Press, Second Edition, 2012.

[5] San Miguel Tello, A. (2017). *Aprendizaje por Refuerzo y Planificación en un Sistema Multi-robot.* Trabajo de Fin de Grado. Universidad de Zaragoza.

[6] Laurent, G. (2002). *Synthèse de comportements par apprentissages par renforcement parallèle.* Tesis Doctoral. Université de Franche-Comté.
Disponible en : https://tel.archives-ouvertes.fr/tel-00008761

[7] Simões, D.L. (2016). *Deep Learning Methods for Reinforcement Learning.* Trabajo de Fin de Máster. Instituto Superior Técnico de Lisboa.

[8] Gonzalez R., Mahulea C., Kloetzer M. (2015). "A Matlab-based Interactive Simulator for Mobile Robotics". *IEEE Conference on Automation Science and Engineering*, Gothenburg, Sweden 24-28 Agosto 2015. Pp 310-315.

Índice de figuras

Apéndice A. Mejora relativa

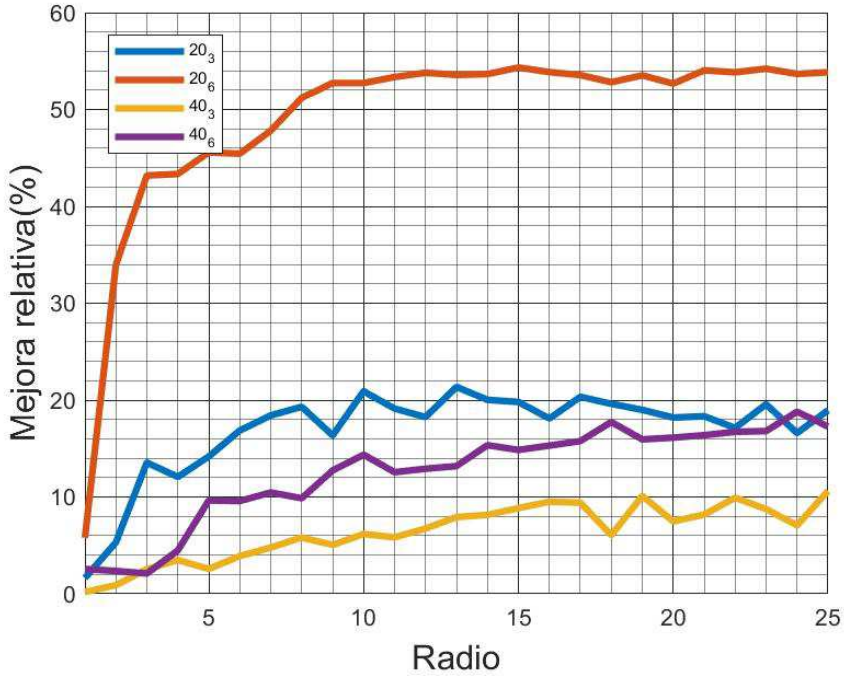


Figura 15. Mejora relativa Protocolo 1.

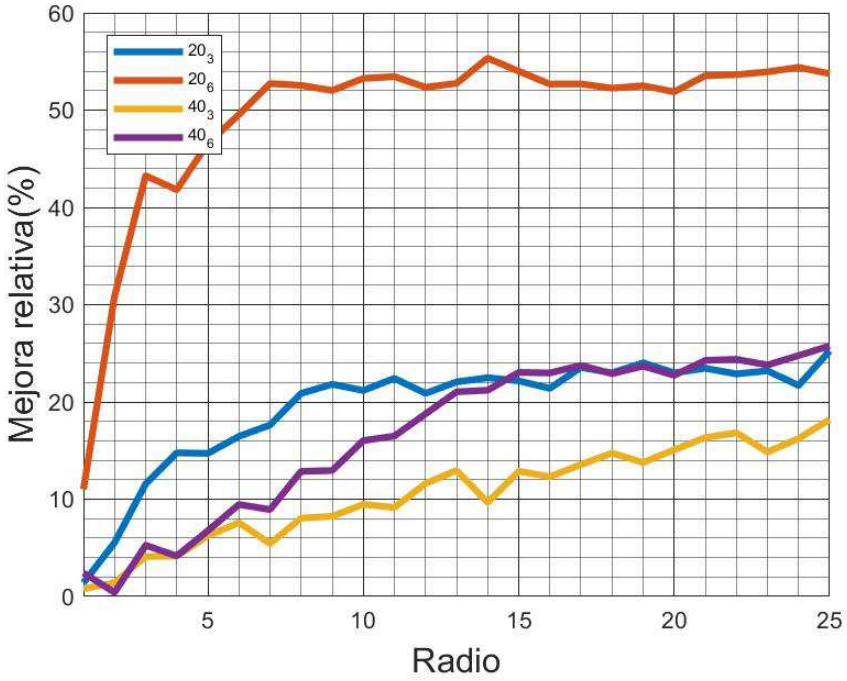


Figura 16. Mejora relativa Protocolo 2.

Apéndice B. Mejora absoluta

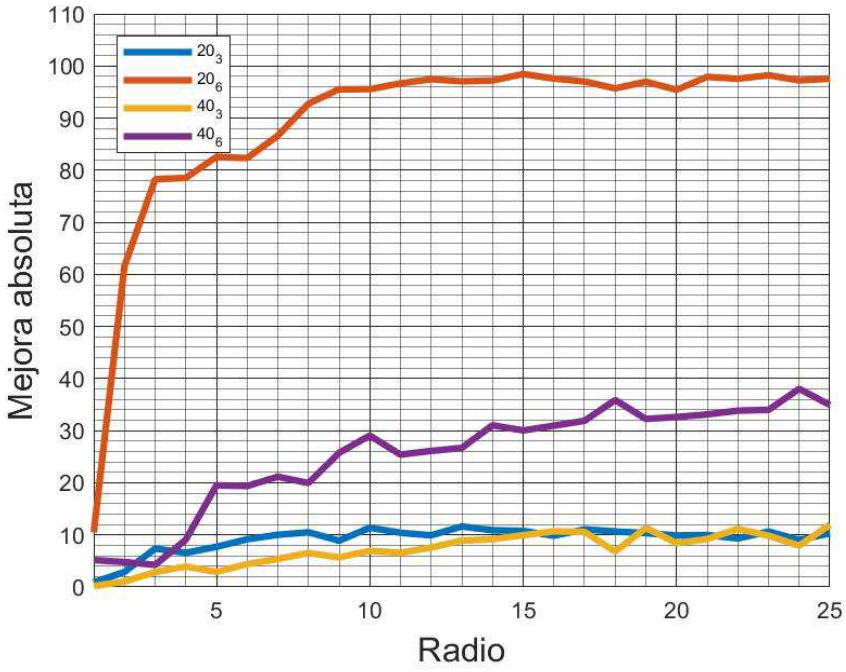


Figura 17. Mejora absoluta Protocolo 1.

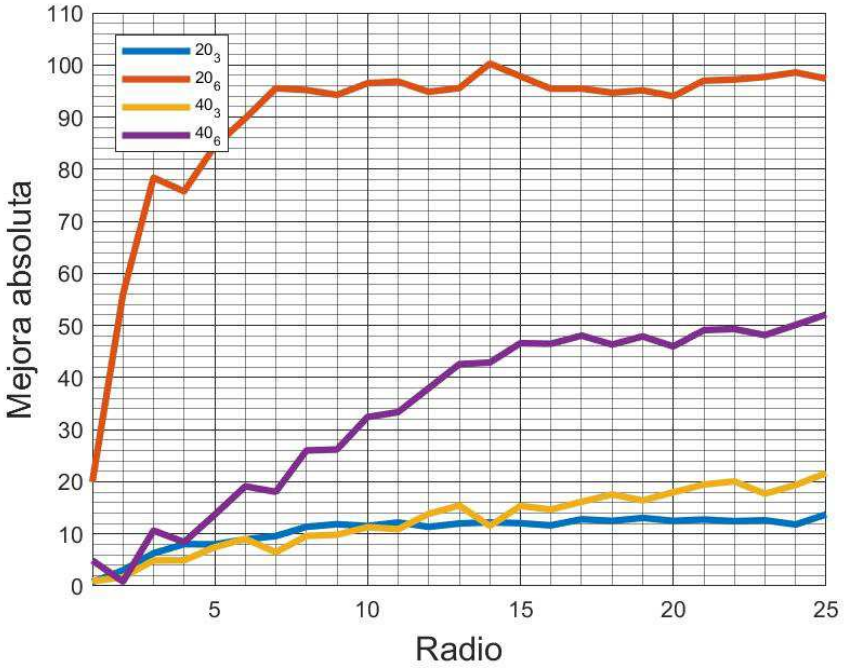


Figura 18. Mejora absoluta Protocolo 2.

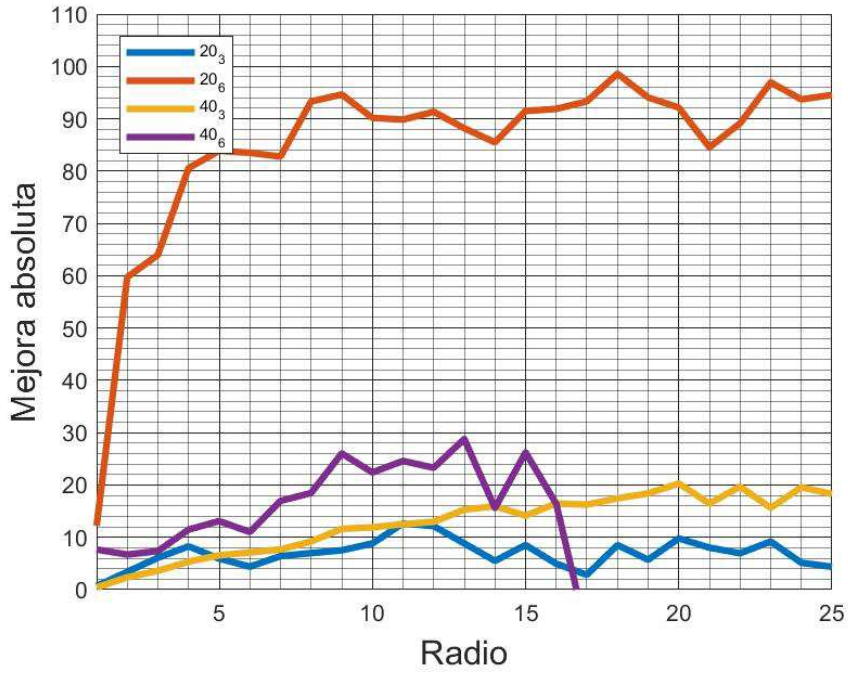


Figura 19. Mejora absoluta Protocolo 4.