Universidad Zaragoza
1542

Trabajo Fin de Grado

Diseño y evaluación de algoritmos de planificación de trayectorias en sistemas multirobot.
Design and evaluation of path planning algorithms for multirobot systems.

Autor/es

Lorenzo Cano Andrés

Director/es

Cristian Mahulea

Escuela de Ingeniería y Arquitectura

2020

CONTENIDO

Resumen

En este trabajo se van a comparar tres algoritmos de generación de trayectorias para equipos de robots móviles. Para ello se va a emplear el paquete de Matlab "Robot Motion Toolbox", desde ahora RMT. Las tres alternativas son variantes dentro del mismo grupo de algoritmos, que parten de un entorno que ha sido discretizado en celdas. Se explicará este proceso de discretización, pero no se profundizará en el tema.

Los tres algoritmos estudiados generan un sistema discreto para representar el entorno y posteriormente emplean algoritmos de búsqueda en grafos o programación matemática para generar las trayectorias. En el primer algoritmo se empleará un autómata finito determinista, mientras que en el segundo y tercero se empleará una red de Petri. La diferencia entre el segundo y el tercero radica en la forma de encontrar una trayectoria en dicha red de Petri. Estos algoritmos serán explicados con más detalle.

La comparación se realizará planteando diferentes escenarios y especificaciones, y midiendo el tiempo que cada alternativa necesita para encontrar una trayectoria que satisfaga dichas especificaciones.

Todas las pruebas se realizarán en el siguiente equipo:

- SO: Windows 10 Home
- Programa: Matlab R2019a
- Placa base: G1.Sniper Z97
- CPU: Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz, 3501 Mhz
- GPU: NVIDIA GeForce GTX 970
- RAM: DDR3 2400 PC3-19200 8GB 2x4GB

1. INTRODUCCIÓN

El campo de los robots móviles autónomos (AMR) dentro de la industria ha crecido continuamente durante los últimos años, y continúa teniendo muy buenas perspectivas de crecimiento.[1]

A pesar de este crecimiento, la gran mayoría de los robots móviles que encontramos en instalaciones industriales son los llamados "vehículos guiados automatizados" (AGV), que siguen caminos prestablecidos en la planta. Normalmente realizan labores automatizadas de transporte. Este tipo de robots, si bien enormemente convenientes en las actividades que realizan, padecen de una importante carencia de flexibilidad.

Los AMR difieren de los AGV en su mayor nivel de autonomía, pues no requieren de trazados preestablecidos. Además, pueden adaptarse con mayor facilidad a diferentes necesidades, ya que no es necesario modificar ningún elemento físico del entorno en el que operan.

No es difícil imaginar ejemplos en los que un equipo de robots móviles autónomos podría automatizar industrias que en la actualidad son inseparables de la mano de obra humana. Un ejemplo podrían ser talleres de mecanizado CNC en los que los operarios tienen la labor principal de mover piezas de una máquina a otra. Con un equipo de robots móviles equipados con brazos robóticos que moviesen piezas y aparejos, podría automatizarse una planta de ese tipo para múltiples productos.

Pero las posibilidades que ofrecen los AMR, no tienen por qué quedarse en el entorno industrial. La ausencia de guías físicas permite su uso en exteriores, donde podrían realizar muy diversas labores, desde tareas de inspección de terrenos, a incluso minería. Todo de forma independiente, sin la necesidad de intervención humana.

Actualmente, los mayores problemas en el desarrollo de estos AMR radican en la dificultad de caracterizar el entorno de forma que se puedan expresar los requerimientos. Es decir, generar un modelo del entorno sobre el cual se puedan imponer las restricciones que llevarían al cumplimiento de la tarea asignada.

Una alternativa para lograr esto, consiste en discretizar el entorno, y sobre este entorno discretizado, crear una especificación. Es decir, a partir del entorno real, que puede ser capturado, por ejemplo, con una cámara, se genera una representación discreta de dicho entorno. Y sobre dicha discretización se imponen todas las restricciones y requerimientos que se deben cumplir para ejecutar la tarea.

Es importante remarcar que lo que se busca es no tener que programar cada robot de forma independiente. El ideal al que se aspira es que, teniendo un equipo de robots, un entorno y una tarea a completar; se genere una representación discretizada del entorno,

[1] http://usecim.net/2020/02/20/innovaciones-tecnologicas-y-el-crecimiento-de-los-robots-autonomos-moviles/

sobre esta representación, se imponga una especificación que representa la tarea a realizar, y que se generen las trayectorias para cada robot que cumplan la especificación, de forma que se realice la tarea.

2. JUSTIFICACIÓN DEL TEMA ELEGIDO

En la literatura encontramos muchos estudios en el análisis y diseño de sistemas de eventos discretos. Por ello, resulta muy atractivo el poder emplear estas herramientas para establecer trayectorias que consigan completar las especificaciones requeridas de un equipo de robots.

Cada uno de los algoritmos que se van a estudiar tiene una serie de ventajas y desventajas respecto a los demás, pero no existe una cuantificación respecto a estas diferencias. Por ello, puede resultar conveniente estudiar como varía el tiempo de cálculo de trayectorias en función del número de robots, la complejidad del entorno y la complejidad de las especificaciones, para cada uno de los algoritmos.

3. DESCRIPCIÓN DE LA PROPUESTA

El objetivo final de este trabajo es cuantificar las capacidades de tres algoritmos diferentes, para generar trayectorias en un entorno discretizado. Estos algoritmos partirán, en cada caso, del mismo entorno discretizado, y de una tarea que un equipo de robots móviles tiene que realizar. La tarea en cuestión será expresada a través de una formula LTL (Linear Temporal Logic).

Para cada caso se obtendrán diferentes tiempos de cálculo de cada una de las fases en las que se divide la generación de la trayectoria.

Una vez se obtengan los tiempos para cada una de las diferentes pruebas, se podrá evaluar, para cada algoritmo, como afectan diferentes parámetros a su tiempo de ejecución. Se prestará especial atención a tres parámetros; el número de robots, el número de regiones de interés, y la complejidad de la especificación.

4. MARCO TEÓRICO

Las partes principales en las que se dividen estos tres algoritmos son:

1. Discretización del entorno.

2. Obtención de un modelo discreto del equipo y entorno, y otro modelo discreto de la especificación.

3. Análisis de sistemas discretos para encontrar una trayectoria.

A continuación, se explica de forma breve en qué consisten las diferentes fases.

4.1. Discretización del entorno

La discretización del entorno se lleva a cabo mediante un método de descomposición en celdas. En este caso, se divide el entorno en celdas triangulares.

El entorno se plantea como una zona de trabajo rectangular, en la cual hay una serie de regiones de interés. Estas regiones de interés tienen la característica impuesta de ser polígonos convexos y pueden superponerse los unos con otros.

Al aplicar el algoritmo de descomposición en celdas, el entorno pasa a estar compuesto por una serie de triángulos, de forma que se cumplen las siguientes características:

1. Las celdas adyacentes comparten los lados íntegramente. Es decir, el lado de una celda triangular corresponde al lado de otra. No puede haber un lado en contacto con otros dos, solo con uno más.

2. Los lados de las regiones coinciden con lados de las celdas triangulares.

De forma que se transforma un entorno como este:

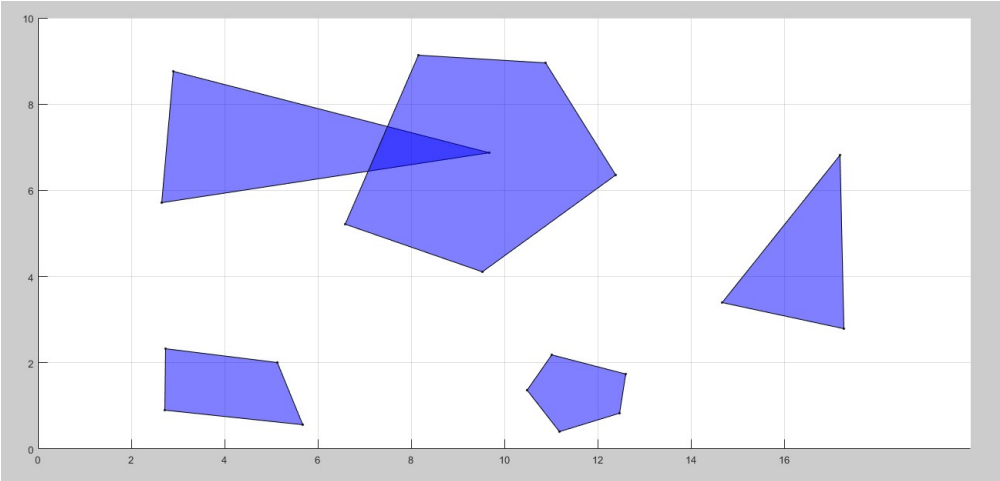

Figura 1: Ejemplo de entorno de trabajo compuesto por 5 regiones de interés

En otro como este:

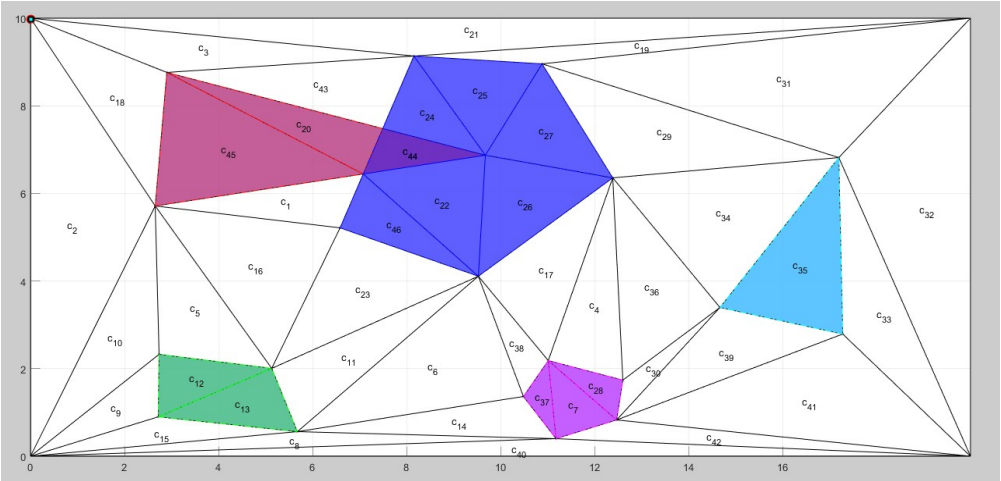

Figura 2: Entorno presentado en la figura 1 particionado en celdas triangulares

Durante este proceso, además, se asigna a cada región de interés una salida determinada. Estas salidas toman el formato de y1, y2….

Las salidas se emplearán posteriormente para indicar la presencia de robots en las diferentes regiones. Si hay un robot en cualquiera de los triángulos de una región de interés, la salida asignada a dicha región se activará. Y en el caso de que un robot esté en una celda que es parte de 2 o más regiones de interés, las salidas de todas estas regiones estarán activas.

En resumen, del entorno se obtendrán una serie de celdas triangulares, cada una con una, varias o ninguna salida asignadas. Asimismo, se obtendrán una serie de relaciones de adyacencia. Esta información se usará posteriormente para modelizar el entorno.

4.2. Generación de los modelos discretos

En este paso encontramos la principal diferencia entre los 3 métodos, que es el formalismo de modelado que emplean. De forma generalizada:

- Método 1: Emplea un autómata finito determinista para modelar el equipo, y un autómata de Buchi para representar la especificación. Una vez generados, se hace el producto entre ambos, obteniendo un único autómata finito determinista.

- Método 2: Emplea una red de Petri para modelar el equipo y un autómata Buchi para modelar la especificación.

- Método 3: Emplea una red de Petri que es el producto de la red de Petri que modela el entorno y el autómata de Buchi que modela la especificación.

Conviene destacar que los autómatas de Buchi de los 3 métodos son iguales, dado que este se genera a partir de la misma especificación.

4.2.1. Autómata finito determinista

Este autómata está compuesto por un set de estados y un set de transiciones.

El estado actual nos indica la celda en la que se encuentra cada robot. El set de estados está compuesto por todas las posibles combinaciones de emplazamientos de los robots en las distintas celdas en las que se ha dividido el entorno. Por ejemplo, si hay 4 celdas y un robot, el autómata tiene 4 estados. En caso de que haya 2 robots, el número de estados del autómata que modela el equipo asciende a 16. De forma que el número de estados aumenta de forma exponencial con el número de robots.

El set de transiciones está compuesto por todas las posibles transiciones de un estado a otro, incluida pasar de un estado al mismo estado.

Cada estado tiene asignada una salida, que es la unión de las salidas activas por la presencia de un robot en las celdas correspondientes.

4.2.2. Red de Petri

En la red de Petri se modela el equipo con un lugar por cada celda. El número de marcas en un lugar representa el número de robots que se encuentran dentro de esa celda.

Las transiciones de esta red de Petri se disparan cada vez que un robot cruza una frontera entre dos celdas.

Cada lugar tiene asignada una serie de salidas, dependiendo del número de regiones de interés asociadas. Si el marcado en un lugar que representa una región(es) de interés es mayor a 0, las salidas asociadas estarán activas.

4.2.3. Autómata de Buchi

El autómata de Buchi se crea a partir de la formula LTL correspondiente a la especificación.

La entrada de este autómata es una serie infinita de las salidas del sistema discreto que representa el entorno.

De tal forma que en este grafo se pasa de un estado a otro cuando hay un cambio en la salida del equipo de robots. Existen uno o varios nodos finales. Estos nodos finales son a los que hay que llegar para que se cumpla la especificación. Sin embargo, no es suficiente llegar. Es necesario poder ir del nodo final al mismo nodo final infinitas veces, una vez alcanzado.

De esta forma, si se encuentra un camino en Buchi del estado inicial a un estado final, y de dicho estado final a sí mismo, esto implica que se ha encontrado una serie de salidas del sistema que permiten cumplir la especificación.

Es importante tener en cuenta que el que exista un camino en Buchi no implica la existencia de una trayectoria que pueda generar las salidas necesarias.

4.3. Búsqueda de una trayectoria

Una vez generados los modelos discretos, se busca en estos un camino que permita cumplir la especificación. El cómo se busca este camino depende del tipo de grafo en cuestión.

En el método 1 se genera un grafo que es producto del autómata de Buchi y del autómata finito determinista que modela el equipo. Sobre este grafo producto se emplean métodos de búsqueda de un camino de coste mínimo en grafos para encontrar una trayectoria.

En el caso del método 2, primero se busca en el autómata de Buchi un camino desde el estado inicial al estado final. Para cada transición de este camino se comprueba que la transición en concreto puede realizarse, pues recordemos que las entradas de Buchi son las salidas de la red de Petri, y algunas entradas que podría aceptar Buchi pueden no ser generables por la red de Petri. Encontrar un camino en Buchi implica también haber encontrado una secuencia de salidas que permiten cumplir la especificación.

En el método 3, se obtienen las trayectorias resolviendo un problema de optimización a partir de la red de Petri resultante del producto del autómata de Buchi y la red de Petri que modela el equipo. La resolución de este problema es una de las posibles trayectorias.

Más detalles sobre este apartado pueden encontrarse en: (Mahulea, Kloetzer, & Gonzalez, 2020).

4.4. LTL: Linear Temporal Logic

LTL es el formalismo a través del cual se expresan las especificaciones para el equipo. Las fórmulas LTL combinan los operadores lógicos con operadores temporales.

Los operadores lógicos son:

- *And*: “&”
- *Or*: “|”
- *Not*: “!”
- *Implication*: →
- *Equivalence*: ↔

Los operadores temporales son:

- *Eventually*: “F”. La expresión que sigue tiene que ser cierta en algún momento.
- *Allways*: “G”. La expresión que sigue tiene que ser cierta siempre.

– *Until*: “U”. La expresión que precede tiene que ser cierta hasta que la expresión que le sigue sea cierta. Momento en el que la expresión precedente ha de ser falsa.

5. DESARROLLO DE LA PROPUESTA

El desarrollo de este trabajo se ha dividido principalmente en 3 etapas.

5.1. Recopilación de información

Para poder dar contexto a todos los algoritmos, se ha realizado una investigación en cuanto a su funcionamiento. En total se han dedicado alrededor de 30 horas a esta investigación.

5.2. Diseño y realización de los experimentos

Se trata de la parte más larga de este trabajo.

Durante el proceso de diseño de los experimentos se ha tratado de establecer una forma de evaluar los 3 parámetros de interés en distintos contextos.

La estructura de los experimentos es la siguiente:

- En primer lugar, tenemos el tipo de experimento, que tiene relación con las especificaciones que se van a emplear. Concretamente se han usado 3 tipos de experimentos. De “alcanzabilidad”, “alcanzabilidad evitando obstáculos” y “exploración”.

- En segundo lugar, dentro de cada tipología, encontramos distintos entornos, con un numero de regiones diferentes.

- Para cada uno de estos entornos, se han realizado experimentos con diferentes números de robots.

- Por último, para cada uno de estos números de robots, se han empleado una serie de especificaciones.

De esta forma se puede aislar la influencia de cada parámetro de los otros. Y se pueden analizar por separado como el número de robots, el número de regiones o la complejidad de la formula afectan al tiempo de generación de la trayectoria.

Pero el principal problema que se ha enfrentado durante esta etapa es que el número de experimentos resulta inmanejable para hacerlos a mano.

Por ello, gran parte del tiempo invertido en el trabajo ha sido para diseñar un script en Python que, para un determinado entorno y número de robots, ha probado diferentes fórmulas para los 3 algoritmos. Además, ha extraído los tiempos, y los ha formateado para poder crear una base de datos en Excel de la cual extraer información útil.

Este script ha permitido además evitar el fallo humano, dado que el número de experimentos supera los 800.

Es necesario destacar que, debido a las diferencias entre los algoritmos, no se han podido realizar todos los experimentos para todos los algoritmos. Esto es debido a que las necesidades de tiempo en ocasiones llegaban a superar las 10 horas, o sencillamente no había suficiente memoria en el computador.

Además, hay una serie de especificaciones para las cuales no siempre se han podido encontrar trayectorias con los 3 métodos.

En total, esta etapa del trabajo ha llevado aproximadamente 150 horas, con unas 20 horas de trabajo autónomo por parte del script de automatización.

5.3. Extracción de conclusiones

Las conclusiones se han extraído tanto de las observaciones realizadas durante los experimentos, como de los datos numéricos recogidos de estos.

A continuación, se recogerán todas las observaciones dignas de mención que se han realizado a lo largo de todos los experimentos. Por último, en base a estas observaciones, se extraerán conclusiones sobre cómo funciona cada algoritmo. Será de especial interés encontrar los casos de uso en los que cada uno de ellos es superior.

En ocasiones, un algoritmo puede ser superior a otro en base únicamente a los tiempos, pero como veremos, en ocasiones dan resultados finales diferentes a los esperados.

6. TIPOLOGÍA DE EXPERIMENTOS

Se han empleado 3 tipologías.

6.1. "Alcanzabilidad" o "Exploración"

Se plantea un numero de 1 a 7 de regiones de interés. Para cada uno de estos entornos, se han hecho pruebas con un número variable de robots. Este número, para cada entorno, ha ido de 1 al número de regiones de interés. Asimismo, respecto a las especificaciones, para cada entorno se han empleado fórmulas que visitan 1, 2, 3, 4… hasta el número de regiones de interés en el entorno.

El objetivo de este set de pruebas es principalmente que, gracias a la facilidad de la combinatoria que presentan, resulte más sencillo cubrir todas las posibles combinaciones de parámetros.

La fórmula LTL de esta tipología es de la forma Fy1&Fy2….

En la figura 3 encontramos una prueba de "alcanzabilidad" en un entorno de 4 regiones de interés, donde hay 4 robots y en el que la especificación es visitar las 4 regiones.

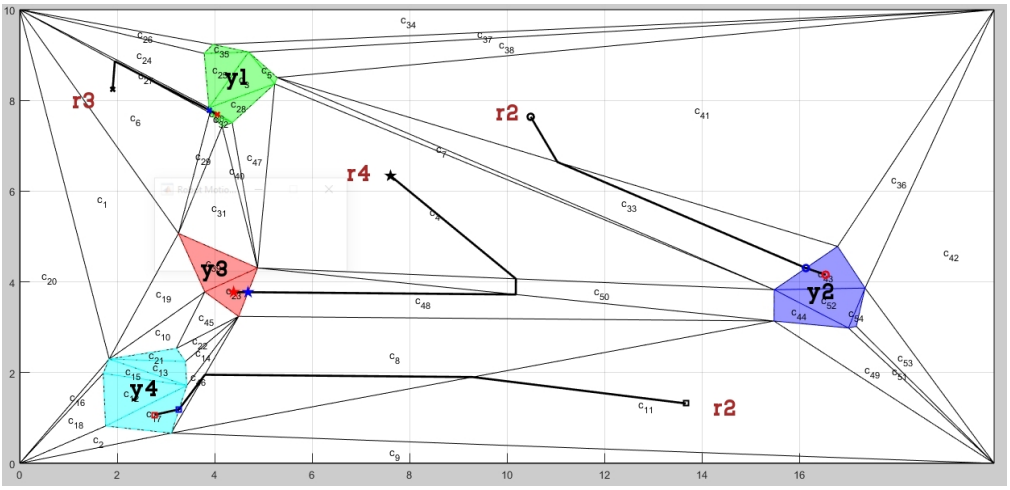

Figura 3. Entorno con 4 regiones de interés y 4 robots. Especificación: Fy1&Fy2&Fy3&Fy4. La trayectoria en la figura corresponde a una posible solución.

6.2. "Alcanzabilidad evitando obstáculos"

En esta serie de pruebas, se van a plantear un conjunto de entornos en los cuales habrá de 2 a 4 regiones de interés y de 1 a 5 obstáculos en el camino de los robots. Esto da un total de 15 entornos diferentes. Asimismo, para cada entorno, el número de robots empleados variará de 1 al número de regiones de interés. También se visitará un numero variable de regiones, desde solamente una, la más alejada, a todas las presentes en el entorno.

El objetivo de esta serie de experimentos es el comprender como el número de obstáculos afecta a los tiempos de cálculo de los métodos. Concretamente, si afecta en la misma medida que el número de regiones a visitar.

Las fórmulas LTL de esta tipología de pruebas son de la forma Fy1&Fy2…&G!(y4|y5…). Las regiones a la izquierda de la G son las regiones a visitar, mientras que las regiones a la derecha de la G son los obstáculos.

En la figura 4 encontramos un ejemplo de este tipo de prueba, en la que hay 4 regiones de interés, 3 obstáculos, 3 robots y la especificación requiere que se visiten las 2 regiones más alejadas.

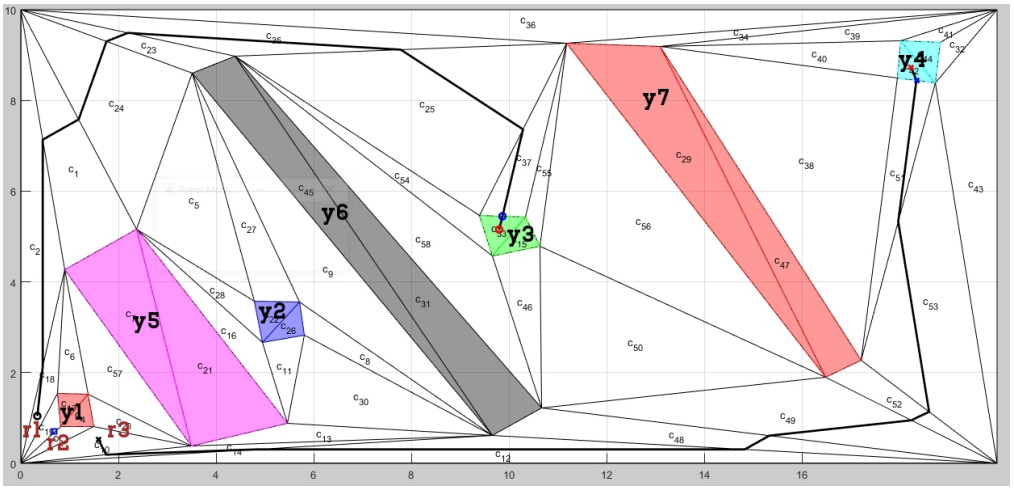

Figura 4. Entorno compuesto por 4 regiones a visitar (pequeñas y poco alargadas) y 3 obstáculos (grandes y alargados), para un total de 7 regiones de interés y 3 robots. Especificación: Fy4&Fy3&(G!(y5|y6|y7)). La trayectoria mostrada corresponde a una posible solución.

6.3. "Vigilancia"

En estas pruebas, al igual que en las de "alcanzabilidad", los entornos se diferencian entre sí únicamente en el número de regiones de interés. Respecto al número de robots, solo se van a emplear o bien 1 o 2 robots. Y solo va a existir una especificación para cada entorno.

La tarea a realizar por los robots es visitar todas las regiones presentes en el entorno de forma cíclica, de forma que, una vez se visiten todas, se vuelva a la primera, y se visiten todas una vez más, repitiéndose este ciclo de forma continua.

La forma característica de la formula LTL de estas pruebas es G(Fy1&Fy2…). Es decir, que siempre se estén visitando todas las regiones dentro de los paréntesis. La figura 5 muestra un ejemplo de tarea de "vigilancia" con 2 robots y 6 regiones a visitar.

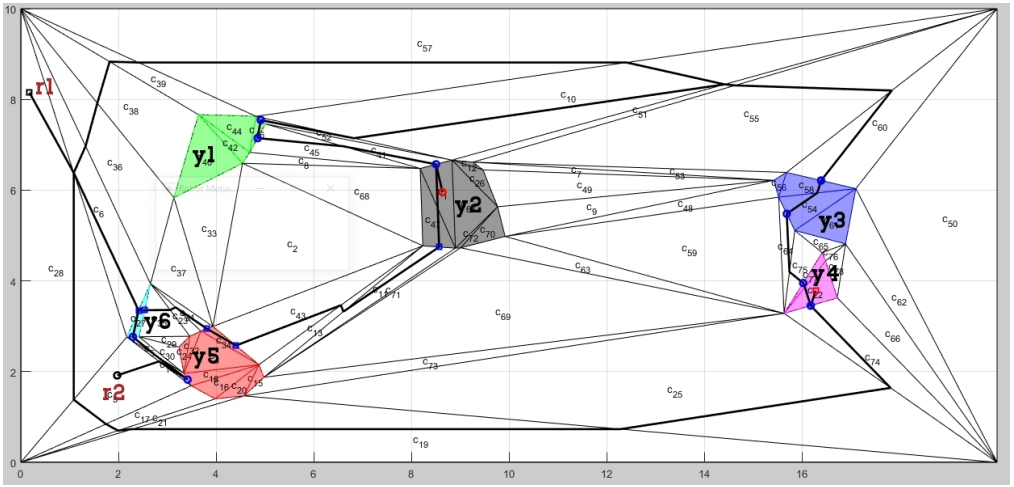

Figura 5. Entorno compuesto por 6 regiones de interés y 2 robots. Especificación: G(Fy1&Fy2&Fy3&Fy4&Fy5&Fy6). La trayectoria mostrada corresponde a una de las posibles soluciones.

7. TRATAMIENTO DE LOS DATOS

La toolbox empleada para realizar las pruebas devuelve tras cada cálculo de trayectoria una serie de tiempos. Estos tiempos son los que el ordenador ha tardado en realizar cada una de las etapas en las que se divide cada método, por lo que varían de un método a otro. De aquí en adelante, a estos tiempos se les referirá de forma abreviada. Por ello, a continuación, se proporciona una breve explicación del significado de cada abreviación.

Para el método que emplea un autómata finito determinista:

- ST T Sis Tran: Tiempo empleado en generar el autómata finito determinista que modela equipo.
- ST T Buchi: Tiempo empleado en generar el autómata de Buchi a partir de la formula LTL.
- ST T Global: Tiempo empleado en generar el sistema de transiciones que combina el autómata finito determinista y el autómata de Buchi.
- ST T Búsqueda: Tiempo que ha costado encontrar una trayectoria en el grafo global.

Para el método que emplea una Red de Petri siguiendo a Buchi:

- PS T Petri: Tiempo empleado en generar la red de Petri que representa el equipo.
- PS T Buchi: Tiempo empleado en generar el autómata de Buchi a partir de la formula LTL.
- PS T Path Buchi: Tiempo que ha costado encontrar un camino en el autómata de Buchi.
- PS T Path: Tiempo que ha costado encontrar un camino en la red de Petri que permita realizar el camino que se ha generado a partir de Buchi.

Para el método que combina la red de Petri y el autómata de Buchi:

- PC T Petri: Tiempo empleado en generar la red de Petri que representa el equipo.
- PC T Buchi: Tiempo empleado en generar el autómata de Buchi a partir de la formula LTL.
- PC T Incluido: Tiempo empleado en generar una red de Petri que combina tanto la red de Petri del entorno como el autómata de Buchi.

- PC T optimización: Tiempo empleado en generar el problema de optimización a resolver para obtener la trayectoria.

- PC T Path: Tiempo empleado en obtener la solución al problema de optimización.

- PC T Proyectar: Tiempo empleado en proyectar la solución obtenida sobre la red de Petri.

8. RESULTADOS

Tras la realización de los experimentos el resultado ha sido una tabla en la que cada fila representa un experimento diferente. En las columnas se representan los parámetros y tiempos obtenidos en cada prueba.

Los parámetros que se tienen en cuenta para clasificar los experimentos son:

- Tipología de la prueba. Puede ser de alcanzabilidad, alcanzabilidad evitando obstáculos o vigilancia.

- Número de prueba de la tipología determinada. Junto a la tipología de la prueba le dan nombre al entorno. Por ejemplo, Alcanzabilidad 3.

- Numero de regiones de interés.

- Numero de robots.

- Formula.

- Método.

Esta es la tabla a partir de la cual se han generado todos los gráficos.

A continuación, se detalla cómo afectan a cada método los distintos parámetros que se han estudiado: número de robots, numero de regiones y formula LTL.

8.1.Autómata finito determinista

El efecto del número de robots es el más pronunciado en este método. Podemos observar en los gráficos del anexo 1 que simplemente pasar de 1 a 2 robots aumenta los tiempos en dos órdenes magnitud. Se puede apreciar que este aumento es principalmente en los tiempos de generación del grafo global (producto del autómata finito que modela el equipo y el autómata de Buchi) y en la búsqueda un camino en este grafo.

El efecto del número de regiones de interés sin embargo es el rasgo que parece tener menor influencia en este método. En el anexo 2 se puede ver con claridad que, para una determinada formula, los tiempos totales permanecen relativamente constantes.

En contraste, se puede observar que la complejidad de la formula afecta a los tiempos de manera más pronunciada que el número de regiones en sí.

8.2. Red de Petri siguiendo a Buchi

El efecto del número de robots en los tiempos de este método parece mínimo. En las gráficas del anexo 2 se puede ver que, a excepción de ciertos picos, las gráficas son relativamente planas con respecto al número de robots.

También parece que, al igual que en el caso del autómata finito determinista, el número de regiones tiene poca importancia. Siendo mucho más influyente la complejidad de la formula LTL. Esta importancia relativa puede apreciarse con claridad en las gráficas del anexo 4.

8.3. Red de Petri con Buchi incluido

El efecto del número de robots en este método, como puede observarse en las gráficas del anexo 2, es considerablemente errático. Pero cabe destacar que, conforme el número de robots se acerca al número de regiones a visitar, el tiempo aumenta. Sin embargo, una vez que el número de robots iguala o supera el número de regiones a visitar, el tiempo disminuye considerablemente.

Respecto al número de regiones y complejidad de las fórmulas se puede observar en las gráficas del anexo 4, que se trata del mismo comportamiento que presentan los otros dos métodos. Sin embargo, cabe destacar que los tiempos que presenta este método son considerablemente superiores a los de Red de Petri siguiendo a Buchi.

9. OBSERVACIONES

En este apartado se detallan algunas observaciones realizadas durante los experimentos que no se muestran claramente en los datos.

Observaciones generales

- Los robots toman los caminos que atraviesan el menor número posible de celdas. Sin embargo, con frecuencia estas trayectorias no coinciden con el camino más corto posible, como se muestra en la figura 6.

- Con frecuencia, ante dos fórmulas que involucran el mismo número de regiones de interés, la fórmula con menos condiciones sobre el orden en el que visitar las regiones es más costosa computacionalmente. Esto es debido principalmente a que una fórmula más restrictiva presenta un autómata de Buchi con menos caminos posibles, y, por lo tanto, más rápido de explorar.

Observaciones sobre el método del autómata finito determinista

- El método del autómata finito determinista es inservible con más de 2 robots en el entorno. No ha sido posible finalizar ningún cálculo de trayectoria que tuviese 3. Y aún con 2, los tiempos de cálculo son muy superiores al resto de métodos.

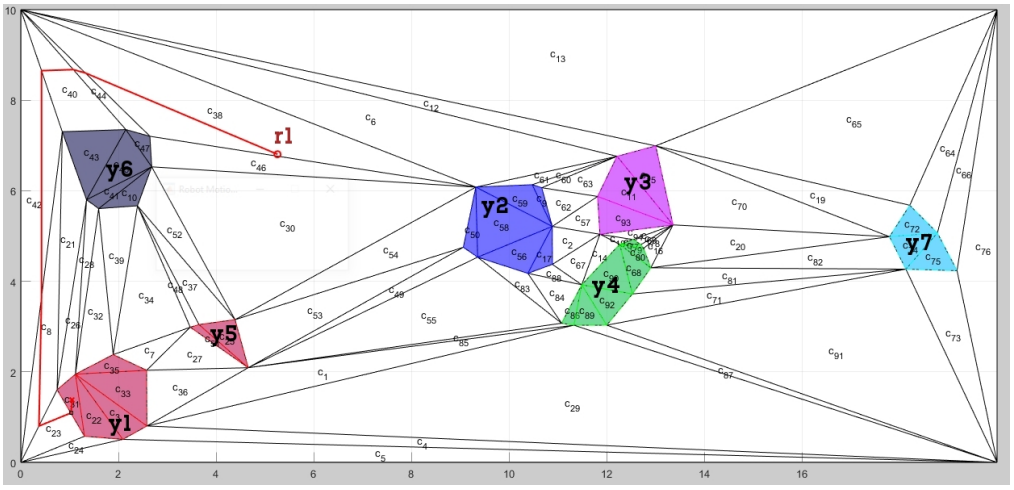

Figura 6. Entorno con 7 regiones de interés y 1 robot. Especificación: Fy1. Se observa que la trayectoria del robot dista de ser la más corta posible.

Observaciones sobre el método de red de Petri con Buchi incluido

- El método de Red de Petri con Buchi incluido presenta el comportamiento más impredecible. Los gráficos del anexo 3 son erráticos, y en los experimentos con 6 o 7 regiones no parece haber correlación entre los diferentes parámetros. Además, las figuras anexo 2: Figura 3 y anexo 4: Figura 3 muestran una anomalía en un experimento en concreto. El de alcanzabilidad, con 6 regiones, 1 robot y especificación de visitar las 6 regiones.

  Esta serie de "incoherencias" indican que este método no está tan directamente relacionado con los parámetros en sí. Podemos tener una formula compleja en un entorno con muchas regiones, pero que, debido al camino de Buchi elegido, el problema de optimización sea resuelto rápidamente.

- El método parece ser especialmente sensible a la relación entre la complejidad de la formula y el número de robots. Concretamente, en el caso de las pruebas de alcanzabilidad, en caso de que el número de robots y el número de regiones a visitar fuesen iguales, o el número de robots fuese superior, el tiempo de resolución es muy inferior a si esto no es así.

  La importancia de esta relación se hace especialmente relevante en los casos en los que hay 6 o 7 regiones. En el anexo 3: Figura 14 se puede observar que, en el caso de visitar 7 regiones, todos los tiempos son 0 (lo que quiere decir que no se ha podido calcular la trayectoria), excepto si el número de robots es 7. Esto se debe a que las pruebas con 6 o menos robots agotaban la memoria disponible en el ordenador. Lo mismo ocurre en el caso de visitar 6 regiones con 5 robots.

- El método de Red de Petri con Buchi incluido es muy sensible al número de marcados intermedios que se emplean en la búsqueda de la trayectoria. En caso de que este número sea demasiado bajo, no será posible encontrar una solución, sin embargo, si este número es muy elevado, la búsqueda de la solución se alarga enormemente.

  Por ello, es importante establecer este número correctamente antes de buscar la trayectoria. Pero esto ha de realizarse en un principio por prueba y error. El tiempo que se ha empleado en esta búsqueda a prueba y error no se incluye en los tiempos finales.

Observaciones sobre implementación

A continuación, se van a comentar observaciones que son relativas a la implementación, y no necesariamente causadas por el algoritmo en sí.

- El algoritmo de red de Petri con Buchi no encuentra soluciones a especificaciones que emplean *Until* (U) junto con *Always* (G).

- El método de red de Petri siguiendo a Buchi no tiene en cuenta las prohibiciones. Como se puede observar en la figura 7, en el entorno de alcanzabilidad evitando obstáculos, se atraviesan los obstáculos.

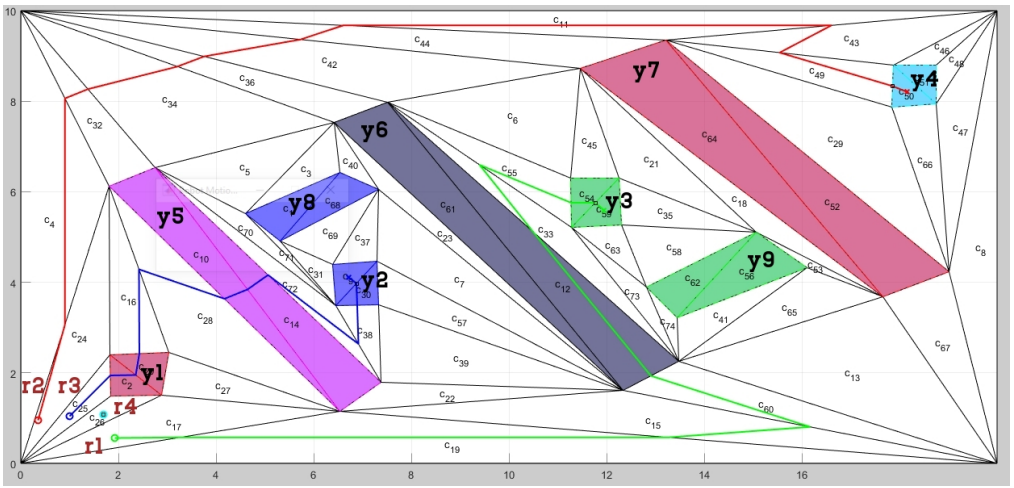

Figura 7: Entorno compuesto por 4 regiones a visitar (pequeñas y poco alargadas) y 5 obstáculos (grandes y alargados), para un total de 9 regiones de interés y 4 robots. Especificación: Fy4&Fy3&Fy2&(G!(y5|y6|y7|y8|y9)). Se puede apreciar que las trayectorias de color azul y verde cruzan los obstáculos.

La causa de puede encontrarse en que, tras encontrar un camino en Buchi, se busca en Petri el camino que proporcione la secuencia de salidas adecuada. Y se acepta cualquier secuencia que la contenga, aunque incluya salidas extra.

10. CONCLUSIONES

Con estos resultados y observaciones complementarias se ha podido establecer un criterio sólido para la elección de un algoritmo adecuado en función de la situación:

10.1. Autómata finito determinista

Se trata, en el caso de 1 robot, del segundo método más rápido, por detrás de Petri siguiendo a Buchi. Sin embargo, el método de Petri siguiendo a Buchi tiene el problema de no ser especialmente fiable para especificaciones complejas.

Por ello, el método del autómata finito determinista es el ideal para casos en los que:

- Hay 1 robot.
- La especificación es compleja, especialmente si contiene regiones prohibidas de forma permanente o circunstancial.

10.2. Red de Petri Siguiendo a Buchi

Este método es, en general, el más rápido de los 3 independientemente del número de robots, complejidad de la formula, o número de regiones. Pero tiene un gran inconveniente que limita su utilidad, la trayectoria que genera no garantiza el cumplimiento exacto de la formula LTL.

Por ello, el caso de uso de este método ha de cumplir:

- La fórmula no puede contener regiones prohibidas.

10.3. Red de Petri con Buchi incluido

Se trata del método más lento, excepto en comparación con el autómata finito determinista con 2 robots. Esto se debe principalmente al tiempo empleado en generar y resolver el problema de optimización.

A esto se le debería sumar el tiempo que cuesta encontrar un número de marcados intermedios adecuado.

Por ello, este método es solamente recomendable en caso de que:

- El número de robots sea 2 o superior.
- La fórmula LTL sea compleja, o difícil de seguir por el método de red de Petri siguiendo a Buchi.

10.4. Tabla resumen

En la tabla a continuación se presenta un criterio de selección donde los números indican la idoneidad de cada método, siendo 2 la máxima idoneidad y 0 la mínima.

Formula	Compleja		Simple	
Robots	1 o 2	Mas de 2	1 o 2	Mas de 2
Autómata finito	2	0	2	0
Red de Petri siguiendo A Buchi	0	1	1	2
Red de petri con Buchi incluido	1	2	0	1

11. APORTACIONES

Durante el desarrollo de este trabajo, gracias al exhaustivo uso al que se ha sometido, se han dado con una serie de formas de mejorar la RMT.

- En primer lugar, la simplificación de los autómatas de Buchi que se generan en los dos métodos que emplean redes de Petri. Esta simplificación consiste en eliminar todos los arcos en Buchi que debido a la limitación en el número de robots no podrían activarse nunca.

  Esta modificación ha llevado a una reducción apreciable en los tiempos de búsqueda de trayectorias en ambos métodos.

- Por las necesidades del trabajo, se ha mejorado la obtención de datos del rendimiento de los algoritmos. Haciendo que los 3 métodos devuelvan los tiempos de cada etapa en la línea de comandos de Matlab.

12. BIBLIOGRAFÍA

Mahulea, C., Kloetzer, M., & Gonzalez, R. (2020). *Path Planning of Cooperative Mobile Robots Using Discrete Event Models.* Wiley-IEEE Press.

13. REFERENCIAS

El enlace a continuación corresponde a un repositorio donde podrá encontrarse tanto una hoja de calculo con todos los tiempos, como el conjunto de todos los entornos, capturas de pantalla y salidas de la consola de Matlab relevantes a este trabajo. También se incluye el script de Python.

https://drive.google.com/drive/folders/1UwY10H46pyXoq7h80sF-98Z_Uu37w2XQ?usp=sharing