Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 18 (2021) 19-31

www.revista-riai.org

Planificación de trayectorias en sistemas multirobot utilizando redes de Petri. Resultados y problemas abiertos

Mahulea, C.[a,*], González, R.[b], Montijano, E.[a], Silva, M.[a]

[a]*Instituto Universitario en Ingeniería de Aragón (I3A), Universidad de Zaragoza, c/ María de Luna 1, 50018 Zaragoza, España.*
[b]*robonity, c/ Axtremadura, A4740 Almería, Aspaña*

To cite this article: Mahulea, C., González, R., Montijano, E., Silva, M., 2021. Path planning of multirobot systems using Petri net models. Results and open problems. Revista Iberoamericana de Automática e Informática Industrial 18, 19-31. https://doi.org/10.4995/riai.2020.13785

Resumen

Este trabajo presenta una estrategia de planificación de trayectorias en equipos de robots móviles basada en el uso de modelos definidos con redes de Petri. Estos tipos de modelos son muy útiles para especificaciones de alto nivel ya que, en este caso, los métodos clásicos de planificación (funciones potenciales, algoritmos RRT, RRT*) no se pueden utilizar, siendo difícil determinar a priori la secuencia de configuraciones para cada robot. Este trabajo presenta la definición formal de la Red de Petri de Movimiento de Robots que se obtiene a partir de una partición del entorno en celdas. Utilizando la estructura de la red de Petri, en caso de especificaciones definidas como fórmulas Booleanas o fórmulas en lógica temporal lineal (LTL), se presentan diferentes problemas de optimización que se pueden utilizar para obtener trayectorias para los robots. La principal ventaja de los modelos basados en redes de Petri es su escalabilidad con respecto al número de robots. Ello permite resolver con eficiencia problemas de planificación de equipos con un número grande de robots. En la segunda parte del trabajo, se presentan algunas extensiones y resultados nuevos para la planificación distribuida en entornos desconocidos y con comunicaciones parciales entre los robots.

Palabras clave: Planificación de trayectorias, sistemas multirobot, sistemas de eventos discretos, redes de Petri.

Path planning of multirobot systems using Petri net models. Results and open problems.

Abstract

This paper presents a trajectory planning approach in multirobot systems based on Petri net models. This type of models is very useful for high-level specifications since, in this case, the classical planning methods (potential functions, RRT algorithms, RRT*) cannot be used being difficult to determine a priori the sequence of configurations for each robot. This work presents the formal definition of the Robot Motion Petri net that is obtained from a partition of the environment in cells. Using the structure of the Petri net, in case of specifications defined as Boolean or Linear Temporal Logic (LTL) formulas, different optimization problems are presented that can be used to obtain trajectories for robots. The main advantage of models based on Petri nets is their scalability with respect to the number of robots. This makes it possible to efficiently solve planning problems with a large number of robots. In the second part of the paper, some extensions and new results for distributed planning in unknown environments and with partial communications between robots are presented.

Keywords: Path planning, multirobot systems, discrete event systems, Petri nets.

*Autor para correspondencia: cmahulea@unizar.es

1. Introducción

Una cuestión fundamental para todo robot móvil es su capacidad para planificar trayectorias libres de colisión. Entre otros muchos casos, estas se pueden definir desde una posición inicial hasta otra final, o la visita programada de una serie de *configuraciones*, es decir, de *regiones* de interés, en un entorno donde existe un conjunto de obstáculos (impedimentos estáticos). Dado un mapa y una posición *objetivo*, la planificación implica determinar una trayectoria que haga que el robot alcance dicha *posición* partiendo de su ubicación inicial (Siegwart and Nourbakhsh, 2004; Choset et al., 2005; LaValle, 2006).

Se pueden identificar cuatro grandes módulos o capas en la arquitectura de navegación típica de un robot móvil (González et al., 2014):

(M1) el nivel más alto es la *planificación de trayectorias*;

(M2) el *controlador de movimientos* (o estrategia de seguimiento de trayectoria). Este es responsable de generar las acciones de control de tal manera que el robot siga con la mayor precisión posible la trayectoria de referencia proporcionada por la capa de planificación (véase, por ejemplo, (Castellanos et al., 2014; Garrido et al., 2013));

(M3) los *controladores de bajo nivel*, que aseguran que las acciones generadas por el controlador de movimientos sean aplicadas por los actuadores del robot (un controlador típico de este nivel es el PID);

(M4) la *localización*, que determina su posición actual.

En lo que sigue se considera el problema de planificación de trayectorias correspondiente al nivel más alto de la arquitectura de un robot móvil. Como es normal en sistemas de control jerarquizado, una observación importante sobre esta arquitectura de navegación tradicional es que el tiempo de ejecución o el tiempo de muestreo de cada capa es diferente, siendo la de planificación de trayectorias la más lenta y las capas (M3) y (M4) las que se ejecutan con mayor frecuencia.

El problema de navegación clásico consiste en alcanzar una configuración final a partir de una inicial, evitando determinados obstáculos. Hay tres grandes áreas en el ámbito de la robótica móvil (Figura 1) (Mahulea et al., 2020a):

(A1) algoritmos combinatorios o exactos de planificación;

(A2) algoritmos de planificación basada en muestreo;

(A3) navegación reactiva.

Los algoritmos pertenecientes a las categorías (A1) y (A2), crean representaciones discretas de un entorno dado (mapa). Los algoritmos combinatorios o exactos (A1), encuentran una solución (si existe) o informan de la inexistencia de una trayectoria (LaValle, 2006). Por el contrario, las soluciones de la segunda aproximación (A2) muestrean escasamente el mapa (normalmente de forma aleatoria), realizando búsquedas discretas que utilizan estas muestras. En este caso se sacrifica la optimalidad absoluta de la ruta (con respecto a la longitud de la trayectoria), siendo su beneficio un cálculo bastante más rápido (LaValle, 2006). Los algoritmos de navegación reactiva no necesitan un mapa del entorno para calcular una trayectoria, mapa que si es necesario en los algoritmos exactos y basados en muestreo.

Un formalismo frecuente para expresar formalmente tareas o especificaciones de alto nivel es la Lógica Temporal Lineal (LTL) (Ding et al., 2014; Guo and Dimarogonas, 2015; Kloetzer and Mahulea, 2015; Ulusoy et al., 2012; Lacerda and Lima, 2019). Mediante fórmulas LTL se pueden expresar tareas más complejas que la navegación clásica; por ejemplo, con especificaciones que requieren movimientos *infinitos* (e.g., en tareas de vigilancia). Las especificaciones LTL pueden aplicarse a un solo robot (Fainekos et al., 2009; Ding et al., 2014; Leahy et al., 2019), definir requisitos individuales para robots móviles de un equipo (Guo and Dimarogonas, 2015) o imponer una especificación global para todo un equipo o grupo robótico, sin asignaciones específicas a estos (Kloetzer and Mahulea, 2015; Ulusoy et al., 2012). Las herramientas y resultados de la teoría de lenguajes formales están involucrados al buscarse una arquitectura de control (Baier and Katoen, 2008). Múltiples enfoques automáticos representan el problema inicial por medio de un modelo de eventos discretos como en los algoritmos del tipo (A1) (Belta et al., 2007; Fainekos et al., 2009; DeCastro et al., 2016). Una solución consiste en obtener el modelo del equipo como producto de N_r autómatas finitos (idénticos), siendo N_r el número de robots. A continuación, se obtiene un autómata Büchi o Rabin correspondiente a la especificación LTL; finalmente, se realiza el producto entre el modelo del equipo con el de la especificación. El modelo resultante permite buscar una ruta aceptada, que luego se proyecta en el modelo del equipo obteniéndose trayectorias para los robots. Esta solución suele conducir al *problema de explosión de estados* (Kloetzer and Mahulea, 2015).

En caso de sistemas multirobot, cuando la especificación de alto nivel concierne al equipo de robots, no hay por el momento una estrategia general asumida por la comunidad de robótica. Se pueden identificar dos posibilidades:

(E1) intentar obtener un modelo global para el equipo y resolver la planificación de forma centralizada;

(E2) estrategias en dos pasos: En el primero se resuelve un problema de asignación de tareas a los robots; posteriormente, se aplica a cada uno un algoritmo de planificación de trayectorias individual (A1), (A2) o (A3).

El problema principal de las estrategias (E2) es que las soluciones no son óptimas en general, ya que las trayectorias se calculan de forma independiente. Por otro lado, el mayor problema de las estrategias (E1) es el tamaño del modelo del equipo que, generalmente, crece de forma exponencial con el número de robots pertenecientes al mismo.

Correspondientes a las estrategias (E2), se puede mencionar la metodología llamada *Asignación y planificación simultánea de tareas* (Schillinger et al., 2018). Se basa en una descomposición de la especificación (Tumova and Dimarogonas, 2016); en particular, la tarea global se descompone en sub-tareas locales que se pueden ejecutar en paralelo. Los algoritmos de planificación basados en la partición en celdas parten de una serie de modelos (típicamente uno para cada robot) que se ejecutan en paralelo, interconectados a través de transiciones especiales que permiten la distribución de la tarea. Una limitación de este enfoque es que no todas las tareas se pueden dividir en sub-tareas

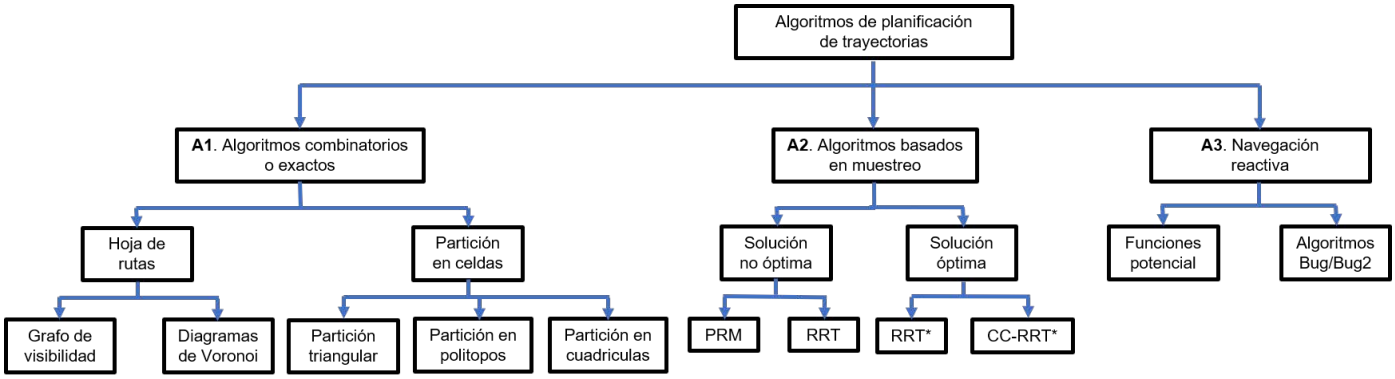

Figura 1: Métodos de planificación de trayectorias para sistemas con un robot para tareas de alcanzabilidad (alcanzar una configuración final) (Mahulea et al., 2020a).

independientes, ya que algunas requieren colaboración y sincronización. Además, el enfoque en (Schillinger et al., 2018) está desarrollado para una clase de tareas donde las trayectorias tienen que ser finitas (fórmulas *co-safe LTL*).

Este trabajo considera una solución alternativa para evitar el problema de explosión del espacio de estados en sistemas multi-robot. La novedad de esta línea proviene de las abstracciones del equipo de robots basadas en redes de Petri (RdP) y de formulaciones en términos de problemas de optimización. Estas herramientas producen una solución computacionalmente manejable, en comparación con los enfoques que utilizan modelos basados en autómatas finitos en los cuales, como se ha mencionado anteriormente, el número de estados crece de forma exponencial. En la Sección 2 se define una *Red de Petri de Movimiento de Robots* (RPMR) que modela a todo el equipo, manteniendo una topología fija, independientemente del número de robots (Kloetzer and Mahulea, 2014). El problema de calcular trayectorias para el equipo se reduce a:

(1) calcular una secuencia de disparo de transiciones para alcanzar un marcado (estado) final deseado; y

(2) producir una observación (salida) requerida que se calcula mediante la resolución de problemas de optimización.

En las Secciones 4 y 5 se describe una solución centralizada en el caso de considerar entornos conocidos. Es decir, se asume que existe un controlador general que determina las trayectorias de los robots.

Asimismo, en la Sección 6, se presentan resultados recientes para el caso de entornos desconocidos y fórmulas Booleanas. Utiliza un algoritmo de inferencia distribuida (Julian et al., 2012), combinado con otro *de consenso* basado en polinomios (Montijano et al., 2013). Se asume que los robots no conocen la ubicación exacta de las regiones de interés. La idea consiste en utilizar el algoritmo de inferencia distribuida para calcular una distribución de probabilidad asociada a cada región. Dado que el cálculo se basa en una iteración de consenso, para reducir el número de rondas de comunicación se propone utilizar el algoritmo presentado en (Montijano et al., 2013), lo que permite explotar su velocidad de convergencia. La combinación de estas técnicas permite al equipo de robots satisfacer la fórmula booleana sin requerir mecanismos complejos de sincronización y/o comunicación.

2. Red de Petri de Movimiento de Robots

Considérese un conjunto de robots idénticos, denominado $R = \{r_1, r_2, \ldots, r_{N_r}\}$, que están operando en un entorno conocido. Por simplicidad, en la presentación de los algoritmos de planificación se abstraerán las capacidades y propiedades de bajo nivel de los robots, asumiendo que disponen de métodos para los problemas (M2)-(M4) que permiten moverlos de un punto a otro. En el entorno existen algunas regiones de interés (no necesariamente disjuntas), etiquetadas con elementos del conjunto $\mathcal{Y} = \{y_1, y_2, \ldots, y_{|\mathcal{Y}|}\}$. Además, se supone que el entorno completo está dividido en un conjunto de regiones disjuntas, llamadas también celdas, por ejemplo, mediante el uso de un método de descomposición de celdas (Choset et al., 2005; Mahulea et al., 2020a). Bajo estos supuestos, el movimiento de los robots se puede caracterizar mediante ejecuciones/transiciones de un sistema discreto.

El conjunto de celdas disjuntas que forma el entorno se denota por $P = \{p_1, p_2, \ldots, p_{|P|}\}$ y el conjunto de etiquetas asignadas a una celda viene dado por una función $h : P \to \mathcal{Y} \cup \{\emptyset\}$. Estas etiquetas corresponden a las regiones de interés, por ejemplo el color de estas regiones. Si la celda p_i está etiquetada por y_j e y_k, entonces $h(p_i) = \{y_j, y_k\}$ mientras que si no hay ninguna etiqueta asociada a p_l, entonces $h(p_l) = \emptyset$. Las regiones etiquetadas con un símbolo no nulo (distinto a $\emptyset$) se llaman regiones de interés.

Ejemplo 2.1. *El entorno presentado en la Figura 2 ha sido particionado en un número de 36 celdas* $P = \{p_1, p_2, \ldots, p_{36}\}$. *En este entorno hay ocho regiones de interés, etiquetadas de la siguiente forma:* $h(p_{26}) = y_{26}$, $h(p_{27}) = y_{27}$, $h(p_{28}) = y_{28}$, $h(p_{29}) = y_{28}$, $h(p_7) = y_7$, $h(p_8) = y_8$, $h(p_{10}) = y_{10}$ *y* $h(p_{11}) = y_{11}$. *Asimismo, existen dos robots idénticos que se encuentran inicialmente en las celdas* p_1 *y* p_5. ■

Considerando todo lo anterior, se abstrae la evolución del equipo robótico a una Red de Petri de Movimiento de Robots (RPMR) definida como:

Definición 2.2. *(Mahulea et al., 2020a) Una Red de Petri de Movimiento de Robots (RPMR) es una tupla* $\mathcal{Q} = \langle \mathcal{N}, \boldsymbol{m}_0, \mathcal{Y}, h \rangle$, *donde:*

- $\mathcal{N} = \langle P, T, \boldsymbol{Post}, \boldsymbol{Pre} \rangle$ *es una red de Petri con:*

- *Un conjunto finito de lugares P (un lugar por cada celda del entorno)*[1]*;*
- *Un conjunto finito de transiciones T; cada una describe la posibilidad de movimiento de un robot entre dos celdas adyacentes;*
- $\boldsymbol{Post} \in \{0,1\}^{|P|\times|T|}$ *es la matriz de post-incidencia definiendo los arcos de transiciones a lugares, tal que* $\boldsymbol{Post}[p,t] = 1$ *si* $t \in T$ *está conectada con el lugar* $p \in P$*, de otra manera* $\boldsymbol{Post}[p,t] = 0$*;*
- $\boldsymbol{Pre} \in \{0,1\}^{|P|\times|T|}$ *es la matriz de pre-incidencia definiendo los arcos de lugares a transiciones, con* $\boldsymbol{Pre}[p,t] = 1$ *si el lugar* $p \in P$ *está conectado con la transición* $t \in T$*, de otra manera* $\boldsymbol{Pre}[p,t] = 0$*;* [2]

- $\boldsymbol{m}_0$ *es el marcado inicial, donde* $\boldsymbol{m}_0[p_i]$ *corresponde al número de robots que se encuentra inicialmente en la celda* $p_i \in P$*;*
- $\mathcal{Y} \cup \{\emptyset\}$ *es el conjunto de los símbolos de salidas;*
- $h : P \to \mathcal{Y} \cup \{\emptyset\}$ *es la función de observación antes definida. Si* p_i *está marcado (i.e., hay un robot en la celda* p_i*), las regiones de interés* $h(p_i)$ *están ocupadas.*

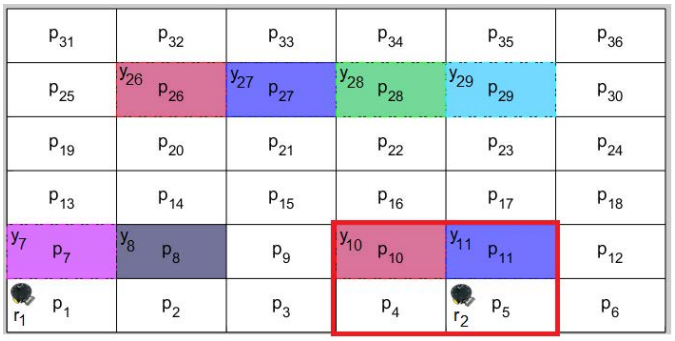

Figura 2: Ejemplo de entorno particionado en celdas $P = \{p_1, p_2, \ldots, p_{36}\}$.

Una RPMR $\mathcal{Q}$ tiene un número de marcas igual al número de robots N_r (cada marca corresponde a un robot), y modela la evolución del equipo al mantener una topología fija, ya que sus lugares y transiciones no cambian al agregar o quitar robots al equipo. Además, cada transición de T tiene solo un lugar de entrada y un lugar de salida, y por lo tanto es una clase particular de redes de Petri llamada *máquina de estados* (Silva, 1985), que no se ha de confundir con los autómatas finitos deterministas ya que la red puede contener varias marcas.

Para una transición $t_j \in T$, ${}^\bullet t_j$ denota su lugar de entrada, mientras que $t_j{}^\bullet$ denota su lugar de salida. Se utilizan análogas anotaciones para los lugares. Formalmente, ${}^\bullet t_j = \{p_i \in P | \boldsymbol{Pre}[p_i, t_j] = 1\}$ y $t_j{}^\bullet = \{p_i \in P | \boldsymbol{Post}[p_i, t_j] = 1\}$. La transición $t_j \in T$ está sensibilizada en el marcado $\boldsymbol{m}$ si su lugar de entrada contiene al menos una marca (es decir, si $\boldsymbol{m}[p_i] \geq 1$, donde $p_i = {}^\bullet t_j$). Una transición sensibilizada t_j se puede disparar (o ejecutar), y la RPMR alcanza un nuevo marcado $\tilde{\boldsymbol{m}} = \boldsymbol{m} + \boldsymbol{C}[\cdot, t_j]$, donde $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ es la matriz de incidencia y $\boldsymbol{C}[\cdot, t_j]$ es su columna correspondiente a t_j.

De acuerdo con la estructura RPMR, el disparo de una transición t_j corresponde al movimiento de un robot desde la celda p_i a la celda p_k. Para el robot en movimiento, la transición t_j significa aplicar una ley de control que conduce al robot de la celda p_i a p_k, y existen trabajos para diseñar tales leyes de control continuas (Habets et al., 2006; Belta and Habets, 2006).

Interesa encontrar secuencias de transiciones que se disparen de manera que el equipo cumpla con una especificación dada. Si se puede alcanzar un marcado $\tilde{\boldsymbol{m}}$ desde $\boldsymbol{m}$ a través de una secuencia finita de disparos de transición σ, se denota con $\vec{\boldsymbol{\sigma}} \in \mathbb{N}_{\geq 0}^{|T|}$ el *vector de disparo* correspondiente: su elemento j^{th} es el número de disparos de la transición t_j en la secuencia σ. En este caso, se cumple la *ecuación de (transición de) estado* o *fundamental* de una red de Petri. Esta viene dada por,

$$\tilde{\boldsymbol{m}} = \boldsymbol{m} + \boldsymbol{C} \cdot \vec{\boldsymbol{\sigma}}. \tag{1}$$

El vector de marcado es el vector de estado de un sistema discreto en el que la matriz dinámica es la identidad; por consiguiente, los lugares de la red de Petri se pueden contemplar como integradores/contadores sometidos a la innovación determinada por el producto de la matriz de incidencia y el vector $\vec{\boldsymbol{\sigma}}$. A pesar de su sencillez, la ecuación fundamental plantea dos problemas importantes: (1) sus soluciones han de ser en los *naturales*, tanto para el vector de marcado como para el de disparo de transiciones; y (2) la existencia de una solución algebraica no significa que exista una secuencia disparable σ que lleve desde el marcado $\boldsymbol{m}$ al marcado $\tilde{\boldsymbol{m}}$. Dicho en términos generales, la complejidad computacional es mayor que la existente en los sistemas lineales y, además, se pueden obtener soluciones *espurias*, lo que puede conducir a semidecisión (Silva et al., 1998). No obstante, en subclases de redes, la ecuación fundamental presenta condiciones necesarias y suficientes para la alcanzabilidad.

Al ser una máquina de estados, para una RPMR viva[3], las soluciones de la ecuación fundamental (1) proporciona el conjunto de marcados alcanzables (Silva et al., 1998). Además, es bien sabido que para esta clase de redes de Petri, si se encuentra un vector de disparo $\vec{\boldsymbol{\sigma}}$ que conduce al modelo a un marcado deseado $\tilde{\boldsymbol{m}}$ disparando el número mínimo de transiciones (es decir, $\vec{\boldsymbol{\sigma}}$ es solución del siguiente problema de optimización: mín $\mathbf{1}^T \cdot \vec{\boldsymbol{\sigma}}$ sujeto a (1)), se puede obtener la secuencia de transiciones correspondiente a los movimientos de los robots (Mahulea et al., 2020a). Sin embargo, si se imponen restricciones en los marcados intermedios (o de manera equivalente sobre $\vec{\boldsymbol{\sigma}}$), esto no es cierto en general debido a los ciclos vacíos (desvíos separados de la trayectoria principal) (Silva et al., 1998). Estos ciclos podrían incluirse en el vector de disparo $\vec{\boldsymbol{\sigma}}$ para satisfacer las restricciones en las marcas intermedias, pero $\vec{\boldsymbol{\sigma}}$ no corresponde a ninguna secuencia de disparo (ver Ejemplo 2.3).

Para cada región de interés y_i, se denota por $\boldsymbol{v}_i \in \{0,1\}^{1\times|P|}$ su vector característico: $\boldsymbol{v}_i[p_k] = 1$ si $h(p_k) = y_i$ y $\boldsymbol{v}_i[p_k] = 0$ en

[1] Para simplificar la notación, se utiliza la misma notación P para el conjunto de celdas y el conjunto de lugares de la red de Petri.
[2] Obsérvese que la clase de redes de Petri aquí considerada es ordinaria, ya que todos los arcos tienen peso unitario.
[3] Una red de Petri es viva si, independientemente del marcado alcanzable, todas las transiciones pueden dispararse en el futuro.

caso contrario, $\forall p_k \in P$. Por tanto, en cualquier marcado $\boldsymbol{m}$, la región y_i está siendo visitada por al menos un robot si $\boldsymbol{v}_i \cdot \boldsymbol{m} > 0$.

La utilización sistemática de técnicas de programación matemática (*Problemas de Programación Lineal*, PPL) en el ámbito de las redes de Petri se introduce en (Silva and Colom, 1988). En el siguiente ejemplo se emplea la variante (2) (en casos posteriores, por ejemplo en (3), se fuerza a que determinadas variables sean enteras, dando lugar a *Problemas de Programación Mixta* (PPM)).

Ejemplo 2.3. *Considérese de nuevo el Ejemplo 2.1. La parte de red de Petri correspondiente al entorno delimitado por el rectángulo rojo en la Figura 2 se representa en la Figura 3. Esta RPMR se compone de cuatro lugares,* $\{p_4, p_5, p_{10}, p_{11}\}$ *correspondientes a las celdas* p_4, p_5, p_{10} *y* p_{11}, *respectivamente. Las transiciones corresponden a la relación de adyacencia. Por ejemplo, debido a que las celdas* p_4 *y* p_5 *son adyacentes, se agregan dos transiciones al modelo,* $t_{4,5}$ *modelando el movimiento de un robot de* p_4 *a* p_5 *y* $t_{5,4}$ *para el movimiento de* p_5 *a* p_4.

Como p_{10} *corresponde a la región de interés etiquetada* y_{10}, *el vector característico de* y_{10}, *denotado* $\boldsymbol{v}_{10}$, *es tal que* $\boldsymbol{v}_{10}[p_{10}] = 1$, *siendo los otros elementos nulos. Del mismo modo,* p_{11} *corresponde a la región de interés etiquetada* y_{11}, *por lo que su vector característico es* $\boldsymbol{v}_{11}[p_{11}] = 1$, *siendo los otros elementos nulos. Supóngase que el robot inicialmente ubicado en* p_5 *debería llegar a* p_{11}. *Se puede obtener un vector de disparo resolviendo el siguiente PPL:*

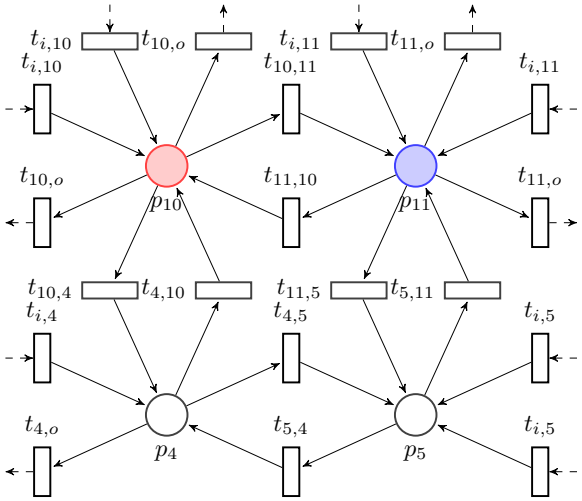

Figura 3: Parte de la RPMR modelando el entorno de la Figura 2.

$$\begin{aligned} &\min_{\vec{\sigma}} \quad \mathbf{1}^T \cdot \vec{\sigma} \\ &\textit{sujeto a} \quad \begin{cases} \boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \vec{\sigma}, \\ \boldsymbol{v}_{11} \cdot \boldsymbol{m} = 1, \\ \boldsymbol{m} \in \mathbb{R}^{|P|}_{\geq 0}, \vec{\sigma} \in \mathbb{R}^{|T|}_{\geq 0}, \end{cases} \end{aligned} \tag{2}$$

donde la primera restricción es la ecuación de estado (1) *que tiene que satisfacer cualquier marcado alcanzable* $\boldsymbol{m}$ *y la segunda restricción es equivalente a* $\boldsymbol{m}[p_{11}] = 1$ *(que haya un robot en el lugar* p_{11} *en el estado (marcado) final* $\boldsymbol{m}$*). Obviamente, la solución de este PPL es un vector de disparos* $\vec{\sigma}$ *que tiene todos los elementos iguales a cero excepto* $\vec{\sigma}[t_{5,11}] = 1$. *En este caso, la secuencia de disparo está compuesta por una sola transición* $t_{5,11}$ *y corresponde al movimiento del robot de* p_5 *a* p_{11}.

Una observación importante sobre el PPL (2) *es que está planteado como un simple problema de programación lineal, que puede devolver soluciones en los racionales. Ahora bien, en una RPMR no tiene sentido hablar de números fraccionarios de disparos, lo que podría plantear un problema. Este no se puede presentar ya que estructuralmente las RPMR son una subclase de* Máquinas de Estados *y para estas (igualmente para sus duales, los* Grafos Marcados*), las matrices de incidencia son unimodulares; en ello radica un mucho menor coste computacional (véase, por ejemplo, (Silva et al., 1998)). Por tanto, para problemas en los que el objetivo es calcular una trayectoria con un marcado final deseado, el PPL* (2) *devuelve una solución entera (Kloetzer and Mahulea, 2014).* ■

Como se ha visto en el ejemplo anterior, para calcular trayectorias para alcanzar algunos estados finales se resuelve un problema de alcanzabilidad en redes de Petri con un coste computacional polinómico. En caso de restricciones sobre la trayectoria, el problema es más complejo pero la solución basada en redes de Petri tiene como ventaja su escalabilidad, como se ilustra en el siguiente ejemplo.

Ejemplo 2.4. *Supóngase ahora que la misión del robot en* p_5 *es nuevamente alcanzar* p_{11}, *pero en su trayectoria ha de pasar por* p_{10} *(correspondiente a la región de interés etiquetada como* y_{10}*). Una primera idea es agregar al PPL* (2) *una nueva restricción que obligue a activar una transición de* ${}^\bullet p_{10}$. *En este caso se tiene un* Problema de Programación Mixta *(PPM) ya que el vector de disparos ha de ser entero (lo que implica que el vector de marcados también lo sea).*

$$\begin{aligned} &\min_{\vec{\sigma}} \quad \mathbf{1}^T \cdot \vec{\sigma} \\ &\textit{sujeto a} \quad \begin{cases} \boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \vec{\sigma}, \\ \boldsymbol{v}_{11} \cdot \boldsymbol{m} = 1, \\ \sum\limits_{t_j \in {}^\bullet p_{10}} \vec{\sigma}[t_j] \geq 1, \\ \boldsymbol{m} \in \mathbb{R}^{|P|}_{\geq 0}, \vec{\sigma} \in \mathbb{N}^{|T|}_{\geq 0}, \end{cases} \end{aligned} \tag{3}$$

con una posible solución $\vec{\sigma}$ *con todos los elementos iguales a cero, excepto* $\vec{\sigma}[t_{5,11}] = \vec{\sigma}[t_{10,4}] = \vec{\sigma}[t_{4,10}] = 1$. *Obviamente, este vector de disparo no puede transformarse en una secuencia de disparo. Esto sucede porque el ciclo vacío* $\{p_4, t_{4,10}, p_{10}, t_{10,4}\}$ *se agrega a la solución asegurando que se creará una marca en* p_{10}, *pero no se puede disparar si no hay por lo menos una marca en* p_4.

Para resolver el problema anterior de vectores de disparo espurios, *en (Mahulea and Kloetzer, 2018) se agregan varios marcados intermedios y en cada paso un robot puede avanzar solo a una celda adyacente. Además, se introducen restricciones adicionales para evitar disparos de ciclos vacíos. En este caso, asumiendo un número de tres marcados intermediarios, el PPM sería:*

$$\begin{array}{ll} \min_{\vec{\sigma}_1,\vec{\sigma}_2,\vec{\sigma}_3} & \mathbf{1}^T \cdot (\vec{\sigma}_1 + \vec{\sigma}_2 + \vec{\sigma}_3) \\ \textit{sujeto a} & \left\{ \begin{array}{l} \boldsymbol{m}_1 = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \vec{\sigma}_1, \\ \boldsymbol{m}_0 - \boldsymbol{Pre} \cdot \vec{\sigma}_1 \geq 0, \\ \boldsymbol{m}_2 = \boldsymbol{m}_1 + \boldsymbol{C} \cdot \vec{\sigma}_2, \\ \boldsymbol{m}_1 - \boldsymbol{Pre} \cdot \vec{\sigma}_2 \geq 0, \\ \boldsymbol{m}_3 = \boldsymbol{m}_2 + \boldsymbol{C} \cdot \vec{\sigma}_3, \\ \boldsymbol{m}_2 - \boldsymbol{Pre} \cdot \vec{\sigma}_3 \geq 0, \\ \boldsymbol{v}_{11} \cdot \boldsymbol{m}_3 = 1, \\ \boldsymbol{v}_{10} \cdot (\boldsymbol{m}_0 + \boldsymbol{m}_1 + \boldsymbol{m}_2) = 1, \\ \boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3 \in \mathbb{R}^{|P|}_{\geq 0}; \vec{\sigma}_1, \vec{\sigma}_2; \vec{\sigma}_3 \in \mathbb{N}^{|T|}_{\geq 0}, \end{array} \right. \end{array} \quad (4)$$

con una posible solución óptima en el cual el $\boldsymbol{m}_1$ tiene una marca en p_4 (i.e., disparando $t_{5,4}$ desde $\boldsymbol{m}_0$); $\boldsymbol{m}_2$ con una marca en p_{10} y cumpliendo con la especificación de la trayectoria (disparando $t_{4,10}$ desde $\boldsymbol{m}_1$); y $\boldsymbol{m}_3$ con una marca en p_{11} y cumpliendo con la especificación del estado final. Otra posible solución (con el mismo coste) corresponde a la secuencia: $t_{5,11}, t_{11,10}, t_{10,11}$. ■

Comentario 2.5. *Si la especificación es compleja y considera no solo el estado final, sino también las trayectorias (por ejemplo, evitando las colisiones entre robots), es necesario introducir marcados intermedios en el problema de optimización. Con ello se aumenta el número de variables e implícitamente la complejidad computacional de su resolución.*

En robótica, el problema de navegación más clásico se circunscribe a alcanzar algunas configuraciones finales de forma óptima. Ampliamente estudiado, se conocen varios algoritmos, normalmente si se considera un único robot. En caso de estrategias A1 (Figura 1), se abstrae el modelo del robot a un grafo (autómata finito) y luego se utiliza un algoritmo de camino mínimo para obtener la trayectoria, como por ejemplo el A^*, D^*, etc. En caso de sistemas multirobot, cuando un conjunto de robots tienen que alcanzar un conjunto de configuraciones finales, el problema es más complicado. Por un lado, si el problema se puede dividir en sub-problemas (para uno o más robots), la solución no suele ser óptima con respeto al equipo. Por otro lado, para obtener un grafo (autómata finito) del equipo, el número de nodos crece de forma exponencial con el número de robots. Eso es porque se requiere hacer el producto síncrono de varios autómatas finitos. Por ejemplo, en caso del entorno de la Figura 2, el modelo de tipo autómata finito del equipo tiene un número de nodos igual a $36^2 = 1296$, siendo (6 x 6) = 36 el número de regiones y 2 el número de robots. Si por ejemplo se quieren considerar un número de 8 robots para alcanzar de forma óptima (minimizando el número de cambios de regiones del equipo) todas las regiones de interés, el número de nodos del autómata finito sería $36^8 = 2,8211 \times 10^{12}$. Evidentemente, explorar este grafo para obtener el camino óptimo puede resultar muy costoso computacionalmente.

Por otro lado, utilizando una RPMR, el modelo del equipo tiene 36 lugares, independiente del número de robots. Además, la complejidad del PPL/PPM no crece al añadir más robots si la especificación se limita a condicionar el marcado final. Solo cambia el estado inicial del problema, añadiéndose únicamente un marcado intermedio adicional para considerar las posibles colisiones. Otra ventaja de las redes de Petri es que permiten obtener trayectorias para problemas más complicados que alcanzar algunas configuraciones finales, como se describen en las dos siguientes secciones.

3. Especificaciones de alto nivel

A continuación se consideran especificaciones más complejas que la simple alcanzabilidad de una configuración final. En particular, la tarea para el movimiento del equipo robot se da como fórmula de una subclase de Lógica Temporal Lineal (LTL) (Clarke et al., 1999; Baier and Katoen, 2008) denotada como LTL_{-X}. Informalmente, una fórmula LTL_{-X} se define recursivamente sobre el conjunto de proposiciones atómicas $\mathcal{Y}$, utilizando los operadores booleanos estándar y algunos operadores temporales con significado directo. La Tabla 1 muestra los operadores booleanos y temporales junto con las constantes que se pueden utilizar para definir tareas para los robots en esta lógica. La principal diferencia de LTL_{-X} con respecto a las fórmulas LTL completas es la ausencia del operador temporal ◯ (siguiente) (Baier and Katoen, 2008). Sin embargo, el operador ◯ tiene sentido solo en caso de sistemas en tiempo discretos y no para sistemas de eventos discretos que se obtienen por la abstracción de un sistema robótico continuo (Belta et al., 2007; Fainekos et al., 2009). Para simplificar la notación, se escribirá LTL en lugar de LTL_{-X}.

Ejemplo 3.1. *La sintaxis mencionada hace que LTL sea muy atractiva para especificar tareas de movimiento. Algunos ejemplos son:*

- Tareas de alcanzabilidad simple*: "llegar a la región y_1 en un estado futuro" escrito como $\phi_1 = \Diamond y_1$,*
- Tareas de alcanzabilidad evitando regiones concretas*: "llegar al estado y_1, evitando siempre y_2", escrito como $\phi_2 = \Diamond y_1 \wedge \Box \neg y_2$,*
- Tareas de convergencia (alcanzabilidad y permanencia)*: "llegar a y_1 y permanecer allí para siempre" - $\phi_3 = \Diamond \Box y_1$,*
- Tareas de vigilancia*: "visitar infinitas veces y_1 y (luego) y_2" - $\phi_4 = \Box \Diamond (y_1 \wedge \Diamond y_2)$,*
- Tareas de alcanzabilidad y elección*: "llegar a la región y_1 o a y_2 $\phi_5 = \Diamond (y_1 \vee y_2)$.*

Además, si hay más de un robot disponible, el logro de regiones disjuntas al mismo tiempo podría ser de interés, como en "llegar a y_1 e y_2 en el futuro"($\phi_6 = \Diamond (y_1 \wedge y_2)$) porque se requiere sincronización entre dos robots para entrar simultáneamente a y_1 e y_2.

Cualquier fórmula LTL sobre el conjunto $\mathcal{Y}$ se puede transformar en un autómata Büchi (Wolper et al., 1983). Las herramientas de software disponibles permiten tales conversiones, por ejemplo, ltl2ba (Gastin and Oddoux, 2001) o Spot (Duret-Lutz et al., 2016).

Definición 3.2. *El autómata Büchi correspondiente a una fórmula LTL sobre el conjunto $\mathcal{Y}$ tiene la estructura $\mathcal{B} = (S, S_0, \Sigma_{\mathcal{B}}, \delta, F)$, donde:*

- *S es un conjunto finito de estados;*

Tabla 1: Constantes, operadores booleanos y temporales utilizados para definir fórmulas *LTL*.

Operadores Booleanos	$\neg$	negación	Constantes	$\top$	Verdadero
	$\vee$	disyunción		$\bot$	Falso
	$\wedge$	conjunción	Operadores temporales	$\mathcal{U}$	until
	$\Rightarrow$	implicación		$\Diamond$	finalmente
	$\Leftrightarrow$	equivalencia		$\Box$	siempre

- $S_0 \subseteq S$ *es el conjunto de estados iniciales;*
- $\Sigma_{\mathcal{B}} = 2^{\mathcal{Y}} \cup \{\emptyset\}$ *es el conjunto de entradas;*
- $\delta : S \times \Sigma_{\mathcal{B}} \to S$ *es la función de transición;*
- $F \subseteq S$ *es el conjunto de estados finales.* ■

Para $s_i, s_j \in S$, se denota con $\rho(s_i, s_j)$ al conjunto de todas las entradas de $\mathcal{B}$ que permiten la transición de s_i a s_j. Las transiciones en $\mathcal{B}$ pueden ser no deterministas, lo que significa que desde un estado dado puede haber múltiples transiciones de salida habilitadas por una misma entrada, i.e., es posible tener $\delta(s, \tau) = \{s', s''\}$, con $s' \neq s''$. Por lo tanto, una secuencia de entrada puede producir más de una secuencia de estados.

Ejemplo 3.3. *Sea la especificación LTL* $\varphi = \Diamond(y_{26} \wedge y_{28}) \wedge \neg(y_{26} \vee y_{28})\mathcal{U}(y_8 \wedge y_{10})$ *para la RPMR de Ejemplo 2.1. La especificación significa que las regiones* y_{26} *e* y_{28} *se deben alcanzar simultáneamente, pero llegando previamente a* y_8 *e* y_{10}*, también de manera simultánea. Un ejemplo de este tipo de problema puede ser un ensamblaje colaborativo en el cual dos robots tienen que llevar sendos objetos de forma simultánea desde las celdas* y_8 *e* y_{10} *y ensamblarlas simultáneamente con otros dos objetos que se encuentran en* y_{26} *e* y_{28}*.*

El autómata Büchi correspondiente a la formula φ *está representado en la Figura 4. De acuerdo con la Def. 3.2,* $S = \{s_0, s_1, s_2\}$ *es el conjunto de estados,* $S_0 = \{s_0\}$ *son los iniciales, y* $F = \{s_2\}$ *los finales; el conjunto de entradas es* $\Sigma_{\mathcal{B}} = 2^{\mathcal{Y}} \cup \{\emptyset\}$ *donde* $\mathcal{Y} = \{y_7, y_8, y_{10}, y_{11}, y_{26}, y_{27}, y_{28}, y_{29}\}$*, y la función de transición* δ *es, por ejemplo,* $\delta(s_1, \{y_{26}, y_{28}\}) = \{s_1, s_2\}$*.*

Una solución que minimiza los movimientos del equipo es la siguiente evolución paralela (Figura 2), para r_1*:* p_1*,* p_2 *donde espera antes de continuar hacia* p_8 *a sincronizarse con el* r_2 *(la fórmula requiere que* p_8 *y* p_{10} *se alcancen simultáneamente); entre tanto,* r_2 *avanza desde* p_5 *a* p_4 *donde espera a la entrada de* p_{10} *a sincronizarse con* r_1*. A continuación, los dos robots entran sincronizados en* p_8 *(*y_8*) y* p_{10} *(*y_{10}*). En este momento, en el autómata Büchi se puede ejecutar la transición* $\delta(s_0, \{y_8, y_{10}\}) = s_1$ *y el estado actual del autómata Büchi se actualiza a* s_1*. A partir de este momento,* r_1 *avanza y sigue de forma secuencial por las siguientes celdas:* p_8*,* p_{14}*,* p_{20} *donde espera sincronizarse con* r_2*. El robot* r_2 *sigue la siguiente secuencia:* p_{10}*,* p_{16}*,* p_{22} *donde espera sincronizarse con* r_1*.*

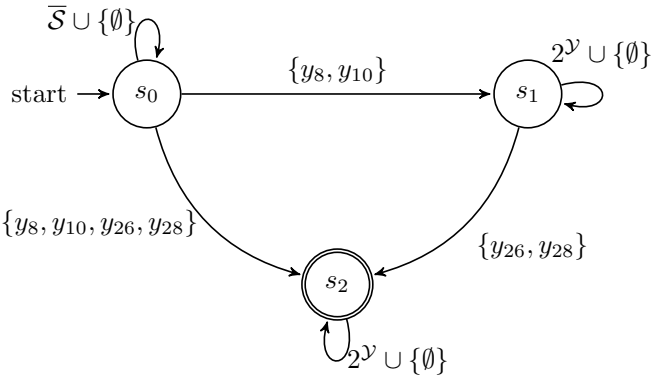

Figura 4: Autómata Büchi correspondiente a la fórmula $\varphi = \Diamond(y_{26} \wedge y_{28}) \wedge \neg(y_{26} \vee y_{28})\mathcal{U}(y_8 \wedge y_{10})$. $\overline{\mathcal{S}}$ contiene todos los conjuntos de $2^{\mathcal{Y}}$ menos los que contienen y_{26} e y_{28}, i.e., $\overline{\mathcal{S}} = \{\mathcal{M} \in 2^{\mathcal{Y}} | y_{26} \notin \mathcal{M} \wedge y_{28} \notin \mathcal{M}\}$.

Los dos robots entraran de forma simultanea en p_{26} *(activando la salida* y_{26}*) y en* p_{28} *(activando la salida* y_{28}*). De esta forma, en Büchi es posible ejecutar la transición* $\delta(s_1, \{y_{26}, y_{28}\}) = s_2$*. Como* s_2 *tiene un autobucle etiquetado con* $2^{\mathcal{Y}} \cup \{\emptyset\}$*, los robots pueden permanecer en las celdas correspondientes.* ■

Una secuencia de entrada (o palabra) infinita con elementos de $\Sigma_{\mathcal{B}}$ es *aceptada* por $\mathcal{B}$ si produce al menos una secuencia de estados (*ejecución*) de $\mathcal{B}$ que visita el conjunto F infinitas veces. En otros términos, una ejecución de $\mathcal{B}$ que visita infinitamente el conjunto F se llama *aceptada*, y si $\mathcal{B}$ tiene al menos una ejecución aceptada, entonces tiene al menos una ejecución con una representación prefijo-sufijo (resultado clásico y elemental que se recoge, por ejemplo, en (Wolper et al., 1983)). Esto significa que la ejecución aceptada (y la palabra de entrada correspondiente) se puede almacenar en la memoria finita en términos de: (i) una secuencia de longitud finita llamada *prefijo* que lleva a $\mathcal{B}$ a un estado final en F, y (ii) una secuencia de longitud finita llamada *sufijo* que lleva a $\mathcal{B}$ de vuelta al estado final alcanzado por el prefijo. La ejecución de $\mathcal{B}$ está formada por el prefijo seguido de infinitas repeticiones del sufijo; es decir, se escribe como { prefijo, sufijo, sufijo, ...}. Por lo tanto, se han de encontrar ejecuciones del modelo de red de Petri con una estructura de prefijo-sufijo similar que genere secuencias de salida que satisfagan las fórmulas LTL.

4. Planificación con especificaciones booleanas

En esta sección se presenta una solución al problema de planificación de trayectorias para especificaciones dadas como fórmulas booleanas (utilizando solo los operadores Booleanos y las constantes en la Tabla 1). Una observación importante es que las trayectorias han de ser finitas en este caso. Asimismo, se

asume que la fórmula está dada en la forma normal conjuntiva (Brown, 2012).

Sea φ la fórmula booleana que los robots deben cumplir en los estados finales. Se define para cada proposición atómica $y_i \in \mathcal{Y}$ una variable booleana x_i tal que $x_i = 1$ si y_i está activa, de lo contrario $x_i = 0$. Además, sea $\boldsymbol{x} \in \{0,1\}^{|\mathcal{Y}|}$ un vector que contenga todas las variables Booleanas x_i. Es posible definir un conjunto de desigualdades lineales usando las variables $\boldsymbol{x}$ de modo que si un vector $\boldsymbol{x}$ es una solución, es decir, satisface las desigualdades, entonces las regiones activas correspondientes a la solución $\boldsymbol{x}$ satisfacen φ[4]. En (Mahulea and Kloetzer, 2018), se presenta un algoritmo tal que para una fórmula booleana dada φ, se calcula el conjunto de desigualdades lineales. Sean estas desigualdades:

$$\boldsymbol{A}_{task} \cdot \boldsymbol{x} \leq \boldsymbol{b}_{task}. \tag{5}$$

Sea $\boldsymbol{\eta} \in \{0,1\}^{|T|}$ el *vector de transiciones a obstáculos*: $\boldsymbol{\eta}[{}^{\bullet}p_k] = 1$ para cualquier p_k que se quiera evitar durante las trayectorias (e.g., los obstáculos), y $\boldsymbol{\eta}[{}^{\bullet}p_k] = 0$ de lo contrario. Básicamente, el vector $\boldsymbol{\eta}$ indica las transiciones que no deberían dispararse para evitar pasar por las regiones no deseadas. El PPM (6) encuentra una secuencia de vectores de disparo $\vec{\boldsymbol{\sigma}}_1$, $\vec{\boldsymbol{\sigma}}_2$, …, $\vec{\boldsymbol{\sigma}}_{N_r+1}$ de modo que la RPMR satisface la fórmula φ en el marcado final $\boldsymbol{m}_{N_r+1}$ evitando durante las trayectorias las regiones no deseadas y los robots no colisionan durante el movimiento.

$$\begin{aligned}
\text{mín} \quad & \mathbf{1}^T \cdot \textstyle\sum_{j=1}^{N_r+1} j \cdot \vec{\boldsymbol{\sigma}}_j \\
\text{sujeto a} \quad & \begin{cases}
\boldsymbol{m}_j = \boldsymbol{m}_{j-1} + \boldsymbol{C} \cdot \vec{\boldsymbol{\sigma}}_j, j = 1,2,\ldots,N_r+1, & (a)\\
\boldsymbol{A}_{task} \cdot \boldsymbol{x} \leq \boldsymbol{b}_{task} & (b)\\
N_r \cdot \boldsymbol{x} \geq \boldsymbol{V} \cdot \boldsymbol{m}_{N_r+1}, & (c)\\
\boldsymbol{x} \leq \boldsymbol{V} \cdot \boldsymbol{m}_{N_r+1}, & (d)\\
\boldsymbol{\eta} \cdot \vec{\boldsymbol{\sigma}}_j = 0,\ j = 1,2,\ldots,N_r & (e)\\
\boldsymbol{Post} \cdot \vec{\boldsymbol{\sigma}}_j + \boldsymbol{m}_{j-1} \leq \mathbf{1},\ j = 1,2,\ldots,N_r+1 & (f)\\
\boldsymbol{m}_{N_r} - \boldsymbol{Pre} \cdot \vec{\boldsymbol{\sigma}}_{N_r+1} \geq \mathbf{0} & (g)\\
\boldsymbol{m}_j \in \mathbb{R}^{|P|}_{\geq 0},\ j = 1,2,\ldots,N_r+1, & \\
\vec{\boldsymbol{\sigma}}_j \in \mathbb{N}^{|T|}_{\geq 0},\ j = 1,2,\ldots,N_r+1, \boldsymbol{x} \in \{0,1\}^{|\mathcal{Y}|}. &
\end{cases}
\end{aligned} \tag{6}$$

El objetivo del PPM (6) es alcanzar un marcado final $\boldsymbol{m}_{N_r+1}$ en el cual la especificación se cumpla (i.e., los robots llegan a unas celdas tal que las salidas activas cumplen con la especificación Booleana). Para garantizar que los robots evolucionan sin colisiones, PPM (6) utiliza $N_r + 1$ marcados intermedios entre $\boldsymbol{m}_0$ y $\boldsymbol{m}_{N_r+1}$. Los robots deben sincronizarse en cada uno; es decir, deben esperar a que el equipo alcance su estado intermedio, para continuar posteriormente el movimiento. Es importante mencionar que es posible que una región que se desea evitar a lo largo de la trayectoria se visite en el marcado (situación) final. Para hacer frente a esto, los robots pueden avanzar evitando colisiones hasta $\boldsymbol{m}_{N_r}$. Desde $\boldsymbol{m}_{N_r}$ a $\boldsymbol{m}_{N_r+1}$ cada robot realiza como máximo un movimiento para cumplir con la especificación sobre el estado final.

La función de coste del PPM (6) que se quiere minimizar es una suma ponderada de los vectores de disparo $\vec{\boldsymbol{\sigma}}_j$. Debido a los pesos, se fuerza a que los últimos vectores de disparo contengan un número menor de transiciones. El objetivo es reducir el número de sincronizaciones. En efecto, si un vector $\vec{\boldsymbol{\sigma}}_j$ resulta nulo, entonces $\boldsymbol{m}_{j-1} = \boldsymbol{m}_j$, lo que significa que habrá menos marcados que sincronizar. Además, debido a la minimización de la función de coste, no se realizarán transiciones innecesarias.

El conjunto de restricciones (6a) corresponde a la ecuación de estado (1) que todos los marcados alcanzables intermedios deben cumplir. La restricción (6b) corresponde a la fórmula que debe cumplirse en el marcado final $\boldsymbol{m}_{N_r+1}$, al imponer condiciones para las variables $\boldsymbol{x}$. Para evitar los obstáculos durante la trayectorias se utilizan las restricciones (6e).

Las desigualdades (6c) y (6d) restringen el marcado final $\boldsymbol{m}_{N_r+1}$ para cumplir con la fórmula booleana correspondiente a los valores de $\boldsymbol{x}$. Básicamente, si se debe visitar una región y_i en $\boldsymbol{m}_{N_r+1}$ para cumplir con la formula booleana, entonces $\boldsymbol{v}_i \cdot \boldsymbol{m}_{N_r+1}$ tiene que ser mayor o igual a 1 (dependiendo de cuantos robots lleguen a las celdas etiquetadas y_i). Esto se impone con la restricción (6d). Por el contrario, si no se debe visitar y_i en $\boldsymbol{m}_{N_r+1}$ y $x_i = 0$, entonces la restricción (6c) impone que $\boldsymbol{v}_i \cdot \boldsymbol{m}_{N_r+1} = \mathbf{0}$.

El conjunto de desigualdades (6f) exige que cada lugar (celda) sea cruzado por un robot como máximo durante los marcados sucesivos $\boldsymbol{m}_{j-1}$ a $\boldsymbol{m}_j$. Esto se debe a que el vector $\boldsymbol{Post} \cdot \vec{\boldsymbol{\sigma}}_j + \boldsymbol{m}_{j-1}$ (con tamaño $|P|$) contiene el número total de visitas a cada lugar en el movimiento de $\boldsymbol{m}_{j-1}$ a $\boldsymbol{m}_j$. Por fin, la restricción (6g) impone que cada robot ejecute como máximo una transición entre $\boldsymbol{m}_{N_r}$ y $\boldsymbol{m}_{N_r+1}$.

El PPM (6) proporciona un método completo para satisfacer una fórmula booleana en el estado final. Los vectores de disparo $\vec{\boldsymbol{\sigma}}_j$ solución de PPM (6) se asignan a secuencias de movimientos de robots. Estos deben sincronizarse en cada marcado intermedio devuelto por PPM (6). De lo contrario, podrían aparecer colisiones entre los robots. Algunas sincronizaciones pueden no aparecer, indicadas por algunos vectores nulos $\vec{\boldsymbol{\sigma}}_j$ de la solución. Más detalles sobre este método se pueden consultar en (Mahulea et al., 2020b) donde adicionalmente es posible añadir una especificación booleana también sobre la trayectoria (y no solo para el estado final).

Ejemplo 4.1. *En el entorno de la Figura 2, para el equipo de los dos robots, se va a considerar la siguiente fórmula Booleana:* $\varphi_1 = y_{26} \wedge y_{28}$ *(i.e., las celdas* p_{26} *y* p_{28} *se deben alcanzar en el estado final) pero durante las trayectorias se quieren evitar las celdas etiquetadas* y_7, y_8, y_{10} *e* y_{11}.

Si se quiere resolver la planificación utilizando el método basado en autómatas finitos (Fainekos et al., 2009) (asumiendo la fórmula LTL: $\varphi_2 = \Diamond y_{26} \wedge \Diamond y_{28} \wedge \Box\neg(y_7 \vee y_8 \vee y_{10} \vee y_{11})$ *que es equivalente a la especificación expresada en la fórmula Booleana* φ_1 *y las restricciones en las trayectorias), el autómata finito tiene un número de* 5184 *estados (*$36^2 = 1296$ *correspondientes al modelo del equipo que multiplicado por* 4*, el número de estados del autómata Büchi). En aproximadamente* $3,5$ *segundos*[5] *se obtienen trayectorias para los robots utilizando la* Robot Motion Toolbox *(RMT) de Matlab (González et al., 2015; Parrilla et al., 2017). No obstante, este método:*

[4] La transformación de fórmulas booleanas en sistemas de desigualdades lineales y en el marco de las redes de Petri fue empleada en (Silva, 1985).

[5] Todos los experimentos se han realizado en un computador Intel i5-6500 @3.2GHz con 16GB RAM.

(1) no garantiza que las trayectorias estén libres de bloqueos y (2) es necesario un paso adicional para eliminar del autómata finito los estados correspondientes a posibles colisiones (por ejemplo estados correspondiente a varios robots en la misma celda). Evidentemente, este post-procesamiento requiere tiempo de cálculo adicional.

Utilizando el método presentado en esta sección (Mahulea et al., 2020b), empleando una RPMR con 36 lugares y 120 transiciones y PPM (6) *en* $0,03$ *segundos se obtiene una solución libre de bloqueos con CPLEX y la RMT. En particular, las secuencias de regiones para los robots son:*

- r_1*:* p_1, p_2, p_3, p_9, p_{15}, p_{21}, p_{27} *y* p_{26}*;*
- r_2*:* p_5, p_6, p_{12}, p_{18}, p_{24}, p_{30}, p_{29} *y* p_{28}.

Las dos trayectorias se pueden ejecutar en paralelo ya que las regiones intermedias son disjuntas. En la solución del PPM (6) *solo el vector de disparo* $\boldsymbol{\sigma}_1$ *es distinto de cero.*

Asumiendo dos robots adicionales, r_3 *en* p_2 *y* r_4 *en* p_4 *y la fórmula Booleana* $\varphi_3 = y_{26} \wedge y_{27} \wedge y_{28} \wedge y_{29}$ *(i.e., las celdas* p_{26} p_{27}, p_{28} *y* p_{29} *se deben alcanzar simultáneamente) pero durante las trayectorias se quieren evitar las mismas celdas de antes (i.e.,* y_8, y_9, y_{10} *e* y_{11}*), el método basado en autómatas finitos requiere trabajar con un grafo cuyo número de nodos es igual a* $36^4 \times 16 = 26{,}873{,}856$*. Evidentemente, la búsqueda en este grafo es mucho más compleja. El método aquí presentado y basado en RdPs devuelve una solución que consiste en dos pasos: primero se mueven los robots* r_2 *y* r_3 *mientras* r_1 *y* r_4 *se quedan parados. Las trayectorias son:*

- r_2*:* p_5, p_6, p_{12}, p_{18}, p_{24}, p_{30}, p_{29} *y* p_{28} *(activando* y_{28}*);*
- r_3*:* p_2, p_3, p_9, p_{15}, p_{21}, p_{27} *y* p_{26} *(activando* y_{26}*).*

En el segundo paso, se mueven los otros robots siguiendo las siguientes trayectorias:

- r_1*:* p_1, p_2, p_3, p_9, p_{15}, p_{21}, p_{27} *(activando* y_{27}*)*
- r_4*:* p_4, p_5, p_6, p_{12}, p_{18}, p_{17}, p_{23} *y* p_{29} *(activando* y_{29}*).* ■

5. Planificación con especificaciones LTL

El objetivo en este caso es elegir una secuencia de observaciones que satisfaga la fórmula LTL, para posteriormente generar disparos que se produzcan en la RPMR. El Algoritmo 1 describe una solución.

Algoritmo 1: Algoritmo de planificación con especificaciones LTL

```
1 Calcular κ caminos en B en forma de prefijo-sufijo
    utilizando el algoritmo k-shortest path (Yen, 1971) y
    guardarlos en Γ;
2 mientras el conjunto Γ no esté vacío hacer
3 |   Elegir una secuencia ∇ de Γ;
4 |   si existe una secuencia de disparos en la RPMR de
  |     modo que las observaciones generadas produzcan
  |     las transiciones de ∇ entonces
5 |   |   Devolver las trayectorias de los robots;
6 |   de lo contrario
7 |   |   Γ = Γ \ ∇;
8 |   fin
9 fin
```

En el primer paso del Algoritmo 1, se construye un conjunto Γ compuesto por κ secuencias aceptadas de $\mathcal{B}$. Estas, con una estructura de prefijo-sufijo en $\mathcal{B}$, se encuentran al ejecutar búsquedas con el algoritmo de k-shortest paths (Yen, 1971) en $\mathcal{B}$, cuya complejidad es $O(\kappa n^3)$, siendo n el número de nodos del grafo. De un estado inicial de $\mathcal{B}$ se encuentran secuencias a estados finales, siendo estas los prefijos. Para cada prefijo encontrado, se construyen secuencias que traerían $\mathcal{B}$ al estado final alcanzado por el prefijo. Cada ejecución agregada en el conjunto Γ incluye el prefijo (que lleva $\mathcal{B}$ a un estado final) y una iteración del sufijo (que lleva a $\mathcal{B}$ a ese estado final). Por lo tanto, cada elemento de Γ es finito, y la longitud infinita necesaria para la semántica LTL surgirá de la repetición infinita del sufijo.

El bucle implementado en los pasos 2 a 9 del Algoritmo 1 se repite hasta que en la RPMR se obtenga una secuencia de marcados que genere una secuencia de observaciones que produzcan las transiciones de la secuencia elegida, o cuando no hay más trayectorias disponible en Γ. Para habilitar la transición $s_j \rightarrow s_{j+1}$ en la secuencia ∇ de Γ se deben cumplir las siguientes condiciones:

(a) El sistema RPMR tiene que alcanzar un marcado final $\boldsymbol{m}$ que genera cualquier observación del conjunto $\rho(s_j, s_{j+1}) \subseteq 2^{\mathcal{Y}} \cup \{\emptyset\}$ que se puede poner como una fórmula Booleana;
(b) Los marcados intermedios de RPMR deberían generar sólo observaciones del conjunto $\rho(s_j, s_j) \subseteq 2^{\mathcal{Y}} \cup \{\emptyset\}$, de modo que desde s_j no se ejecute otra transición en $\mathcal{B}$.

Para calcular un marcado final $\boldsymbol{m}$ en el cual se genere una observación del conjunto $\rho(s_j, s_{j+1})$, es posible utilizar el PPM (6). El conjunto $\rho(s_j, s_{j+1})$ se puede escribir como fórmula booleana con las variables x_i definidas en la sección anterior. Además, para comprobar que en los marcados intermedios no se generan observaciones que resulten en una transición no deseada en $\mathcal{B}$ se pueden utilizar los resultados presentados en (Kloetzer and Mahulea, 2020). En caso contrario, cuando es posible que otra transición en Büchi se ejecute, se considera otro camino de Γ y el proceso se repite.

Es evidente que el algoritmo no es completo porque no garantiza que se obtenga una solución. Si no se obtiene una solución, se puede incrementar κ y obtener más caminos en Büchi. Se ha observado que, casi siempre, con κ pequeños (generalmente $\kappa \leq 4$) se obtiene una solución.

Ejemplo 5.1. *Considérese el escenario de la Figura 2 y la especificación LTL para el equipo de dos robots* $\varphi = \square\diamond((y_{26} \wedge y_{28}) \wedge \diamond(y_8 \wedge y_{10}))$*. Con esta se requiere que las regiones* (y_{26}, y_{28}) *e* (y_8, y_{10}) *se visiten infinitamente. El autómata Büchi* $\mathcal{B}$ *correspondiente a* φ *aparece en la Figura 5. Ejecutando el Alg. 1, algunos caminos devueltos son:* $\nabla_1 = s_0\, s_0$*, con el sufijo* s_0 *que corresponde al camino más corto para visitar infinitamente el estado final* s_0*. Para este camino en Büchi, la RPMR* $\mathcal{Q}$ *se utiliza para encontrar un marcado final en la que se observan* y_{26}, y_{28}, y_8 *e* y_{10}*. Sin embargo, como únicamente se tienen dos robots disponibles, no es posible obtener una solución al PPM* (6)*, ya que es imposible activar las cuatro regiones a la vez. Otro camino en Büchi es* $\nabla_2 = s_0\, s_1\, s_0$ *que tampoco se puede seguir ya que la transición de* s_1 *a* s_0 *requiere las mismas observaciones que antes:* y_{26}, y_{28}, y_8 *e* y_{10}.

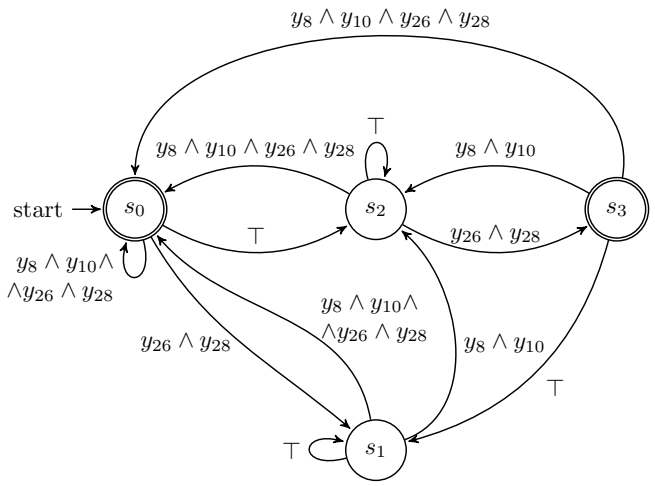

Figura 5: El autómata Büchi correspondiente a la fórmula $\varphi = \square\diamond((y_{26} \wedge y_{28}) \wedge \diamond(y_8 \wedge y_{10}))$.

Finalmente, el camino $\nabla_3 = s_0\, s_2\, s_3\, s_1\, s_2\, s_3$ (con sufijo $s_1\, s_2\, s_3$) se puede seguir por el equipo. En particular, los dos robots evolucionan de la siguiente manera: r_1 desde p_1 va a p_7, p_8, p_{14}, p_{20} y p_{26} y espera sincronizarse con r_2 que en paralelo va a p_4, p_{10}, p_{16}, p_{22} y p_{28}. Cuando llega el primer robot a su celda intermedia, la transición en Büchi $\delta(s_0, \tau) = s_2$ se ejecuta, donde $\tau \in \{y_{26}, y_{28}\}$. Cuando los dos robots están simultáneamente en p_{26} y p_{28} las salidas y_{26} e y_{28} están activas y en Büchi se ejecuta $\delta(s_2, \{y_{26}, y_{28}\}) = s_3$. A partir de este punto, r_1 baja siguiendo la secuencia p_{26}, p_{20}, p_{14} y p_8 donde espera al r_2 que ejecute la secuencia p_{28}, p_{22}, p_{16}, p_{10}. En cuanto el primer robot llega a su nueva posición intermedia, se ejecuta en Büchi $\delta(s_3.\tau) = s_1$ con $\tau \in \{y_8, y_{10}\}$. Cuando ambos robots llegan a sus posiciones finales, y_8 e y_{10} están activas y se ejecuta en Büchi $\delta(s_1, \{y_8, y_{10}\}) = s_2$. Desde este punto, los robots tienen que ir a las misma regiones de antes p_{26} y p_{28} para ejecutar $\delta(s_2, \{y_{26}, y_{28}\}) = s_3$. A lo largo del camino, hay otras salidas que se generan (por ejemplo $\{\emptyset\}$ cuando los robots están en las celdas blancas) pero en Büchi, el estado s_2 se puede mantener sin problemas ya que tiene un autobucle etiquetado con True. ■

6. Planificación distribuida en entornos (parcialmente) desconocidos utilizando redes de Petri

Las soluciones descritas en las secciones anteriores presuponen que los robots disponen de información completa sobre el entorno en el que se realiza la planificación. Asimismo, se asume que existe una unidad central que se encarga de resolver el problema de planificación, conociendo la posición de todos los robots y el valor de salida de las regiones, determinado por la función de observación h. En el momento en el que se relajan estas suposiciones, aparecen complicaciones adicionales en el problema de planificación. Por un lado, la falta de información sobre la especificación del problema, e.g., en operaciones de rescate se puede conocer el mapa pero no la localización exacta de las víctimas, introduce la necesidad de incluir incertidumbres en el modelo, así como de incorporar mecanismos de exploración en la solución del planificador. Por otra parte, la eliminación de una unidad central de planificación hace recomendable la utilización de algoritmos distribuidos en el proceso de fusión de la información, con el objetivo de evitar sobre costes en las comunicaciones para que todos los robots manejen la misma información. En esta sección se describe una propuesta de solución a estos dos problemas basada en la utilización de algoritmos de inferencia y consenso distribuido.

En particular, en la solución planteada en este artículo se asume que los robots conocen el mapa sobre el que se mueven (los lugares P, las transiciones T y los arcos que los conectan en la RPMR) pero desconocen los valores de la función de observación h; es decir, no tienen constancia de la ubicación en el mapa de las regiones de interés. Asimismo, existe una especificación global definida sobre el resultado de la función h sobre el conjunto de regiones.

El equipo tiene que explorar el mapa para identificar las regiones de interés ejecutando acciones locales. Si estos son capaces de comunicarse, los robots intercambiarán su información, que es de naturaleza probabilista. Utilizando algoritmos de consenso distribuido, la estimación del entorno se mejora y las regiones finales deseadas se identifican y alcanzan más rápidamente.

Resultados preliminares en este marco se presentan en las siguientes sub-secciones (Mahulea et al., 2020c). En la Sub-Sección 6.1 se presenta un método de estimación probabilística de las regiones de interés que se puede ejecutar por cada robot. Estas estimaciones se obtienen por medidas propias pero también se asumen comunicaciones entre robots e intercambios de información. La estimación probabilística se utiliza en la Sub-Sección 6.2 para ejecutar el PPM (6) de forma distribuida (cada robot ejecuta de forma independiente este algoritmo). Se observa que la especificación global se cumple. La convergencia de los nuevos algoritmos no está de momento demostrada siendo un problema que se estudiará en el futuro.

6.1. Estimación de las regiones de interés

Supóngase que cada robot $r_i \in R$ va equipado con sensores capaces de proporcionar una medida imprecisa $o_{r_i}(p_j) \in \mathcal{Y}$ para cada celda $p_j \in P$. La medida depende de cada robot y de su posición en el momento de medir.

Puesto que el problema de estimación es independiente para cada lugar del mapa, se describe el algoritmo para una única región, $p \in P$, haciendo notar que la ampliación al mapa completo se realiza mediante su réplica para todos los lugares. Se denota como $\mathbb{P}(y_k)$, la probabilidad de que la región $p \in P$ sea etiquetada como $y_k \in \mathcal{Y}$. Por lo tanto, como objetivo adicional al problema de planificación se incluye la necesidad de obtener una distribución de probabilidad, $\mathbb{P}(\mathcal{Y})$, que considere las medidas obtenidas en el tiempo por todos los robots, teniendo en cuenta que éstos tienen un radio de comunicación limitada.

Para llevar a cabo esta tarea, se explotan las propiedades que tienen los algoritmos de consenso distribuidos en el ámbito de la percepción (Mesbahi and Egerstedt, 2010; Montijano and Sagüés, 2015). En particular, el cálculo se centra en la utilización de un algoritmo de consenso basado en polinomios de Chebyshev (Montijano et al., 2013), aplicado al problema de inferencia distribuida descrito en (Julian et al., 2012). Para la estimación, a cada robot r_i se le aplica la regla de Bayes a partir

de la observación $o_{r_i}(p)$ obtenida en cada instante θ,

$$\mathbb{P}^{r_i}_{\theta}(y_k|o_{r_i}(p)) = \frac{\mathbb{P}^{r_i}_{\theta}(y_k)\cdot\mathbb{P}(o_{r_i}(p)|y_k)}{\sum\limits_{y_\ell\in\mathcal{Y}}\mathbb{P}^{r_i}_{\theta}(y_\ell)\cdot\mathbb{P}(o_{r_i}(p)|y_\ell)}, \tag{7}$$

donde $\mathbb{P}^{r_i}_{\theta}(y_k|o_{r_i}(p))$ representa la probabilidad de que la región tenga la etiqueta y_k, condicionada por la observación obtenida del sensor, $o_{r_i}(p)$. $\mathbb{P}(o_{r_i}(p)|y_\ell)$ es la probabilidad de obtener dicha medida suponiendo la región tuviera la etiqueta y_ℓ. Por un lado, para el cálculo de $\mathbb{P}^{r_i}_{\theta}(y_k|o_{r_i}(p))$ se asume que el etiquetado de las regiones no cambia en el tiempo, lo que hace que la estimación a priori en el instante θ sea precisamente la probabilidad a posteriori en el instante anterior, i.e., $\mathbb{P}^{r_i}_{\theta+1}(y_k) = \mathbb{P}^{r_i}_{\theta}(y_k|o_{r_i}(p)), \forall y_k \in \mathcal{Y}$. Por otro lado, se asume que los robots conocen la calibración de sus sensores, lo que les permite conocer las probabilidades $\mathbb{P}(o_{r_i}(p)|y_\ell)$ para todo $y_\ell \in \mathcal{Y}$.

Es importante notar que (7) no está teniendo en cuenta para nada las estimaciones que hacen otros robots en el proceso de actualización. En (Julian et al., 2012) se propone aproximar la probabilidad combinada de las medidas de todos los robots por su producto, $\mathbb{P}(O(p)|y_k) = \prod\limits_{r\in R}\mathbb{P}(o_r(p)|y_k)$, donde $O(p)$ representa el conjunto de medidas tomadas por el equipo de robots. Una vez obtenida, esta medida reemplaza al término $\mathbb{P}(o_{r_i}(p)|y_k)$ en (7), dando lugar a una estimación más robusta del etiquetado de la región.

Para realizar dicho cálculo utilizando un número reducido de rondas de comunicación entre los robots, se puede utilizar un algoritmo de consenso distribuido (Montijano et al., 2013). Este utiliza una variable, $\boldsymbol{z^i} \in \mathbb{R}^{|\mathcal{Y}|}$, para cada robot $r_i \in R$, para realizar la estimación de $\mathbb{P}(O(p)|y_k)$. Dicha variable se inicializa con el logaritmo de la probabilidad local del sensor del robot para las diferentes etiquetas, $\boldsymbol{z^i}(0) = \log\left(\mathbb{P}^{r_i}_{\theta}(o_{r_i}(p)|y_k)\right)$ y se actualiza de acuerdo a la siguiente iteración lineal,

$$\boldsymbol{z^i}(1) = \frac{1}{T_1(c)}\cdot\sum_{j\in\mathcal{N}_i\cup i} w'_{ij}\cdot\boldsymbol{z^j}(0), \tag{8a}$$

$$\boldsymbol{z^i}(n) = 2\cdot\frac{T_{n-1}(c)}{T_n(c)}\cdot\sum_{j\in\mathcal{N}_i\cup i} w'_{ij}\cdot\boldsymbol{z^j}(n-1) - \frac{T_{n-2}(c)}{T_n(c)}\cdot\boldsymbol{z^i}(n-2), \tag{8b}$$

donde $\mathcal{N}_i \subseteq R$ denota el conjunto de vecinos del robot r_i, i.e., con los que puede comunicarse directamente, y $T_n(c)$ denota el polinomio de Chebyshev de primer tipo de grado n, calculado de forma recurrente por $T_n(c) = 2\cdot T_{n-1}(c)\cdot c - T_{n-2}(c)$. El parámetro $c > 1$ es un parámetro de diseño libre, mientras que $0 < w'_{ij} < c$ denotan los pesos asociados al intercambio de información entre los robots r_i y r_j, asumiendo que $r_j \in \mathcal{N}_i$ (Montijano et al., 2013). Para concluir, el valor de las probabilidades de las observaciones se obtiene localmente por cada robot mediante

$$\mathbb{P}(O(p)|y_k) = \frac{\exp\left(\boldsymbol{x^i}[k](n)\right)^{|R|}}{\sum_{j\in\mathcal{Y}}\exp\left(\boldsymbol{x^i}[j](n)\right)^{|R|}}. \tag{9}$$

6.2. *Planificación adaptativa*

Se denomina *planificación adaptativa* aquella en que la función h (función de observación) va cambiando (se *adapta*) durante la exploración (i.e., la trayectoria) de acuerdo con la actualización de las medidas obtenidas del entorno.

La ejecución del algoritmo de planificación será distribuida, determinando cada robot una planificación que utiliza la estimación probabilística que tiene sobre el entorno. En particular resuelve un PPM (6) y ejecuta la primera acción (avanza a la región adyacente, estrategia de horizonte deslizante) correspondiente a la solución óptima. En el estado inicial, cada robot r_i construye su propia RPM, siendo los modelos de cada robot diferente en la función de observación h. Mediante el algoritmo de consenso, cada uno actualiza las probabilidades del etiquetado de las regiones y construye la función h, eligiendo para cada región la etiqueta con mayor probabilidad, i.e., $h(p) = \text{máx}_k\,\mathbb{P}^{r_i}_{\theta}(y_k)$. De esta manera cada robot resuelve de forma individual una instancia del PPM (6) y obtiene una solución particular. Para dicha solución, el robot únicamente considera la parte del vector de disparo que va ligada a su propio movimiento y ejecuta únicamente la primera acción. En el caso particular de que el PPM (6) no tenga una solución factible, el robot elige un movimiento aleatorio y lo ejecuta. De esta manera, los robots pueden obtener una observación diferente del entorno que mejore su estimación. La planificación se actualiza mediante un proceso iterativo que va dirigiendo el equipo hacia el cumplimiento de la especificación (el algoritmo completo se presenta en (Mahulea et al., 2020c)).

Cada vez que se actualizan las funciones h y los planes, antes de volver a ejecutar una nueva iteración del algoritmo completo, los robots han de esperar a que el conjunto del equipo haya completado sus movimientos. Esto se hace imponiendo un tiempo de espera equivalente al máximo tiempo requerido para moverse de una celda a otra. El número de iteraciones que se aplica el algoritmo determina la probabilidad de que el conjunto acabe en una configuración que satisfaga la especificación. Cuanto mayor sea este número, más probable será que se alcance un resultado aceptable.

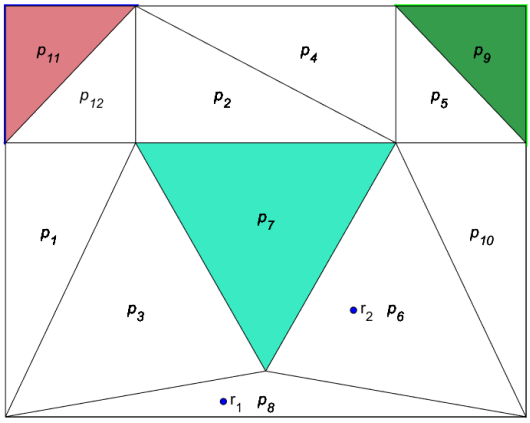

Figura 6: Mapa del ejemplo 6.1.

Ejemplo 6.1. *Se considera el entorno de la Figura 6, que contiene tres regiones de interés (p_7, p_9 y p_{11}) y dos robots inicialmente colocados en p_8 y p_6. El conjunto de observaciones es $\mathcal{Y} = \{y_1, y_2, y_3\}$ y la función de observación real $h(p_{11}) = y_1$; $h(p_9) = y_2$; $h(p_7) = y_3$ con $h(p_i) = \emptyset$ para el resto de regiones. La fórmula booleana a cumplir en el estado final es $\varphi = y_1 \wedge y_2$, lo que significa que un robot debe terminar en p_{11} y otro en p_9.*

La solución que devuelve el PPM (6) *con el entorno conocido mueve a* r_1 *por el camino* $p_8, p_3, p_1, p_{12}, p_{11}$ *y al robot* r_2 *por* p_6, p_{10}, p_5, p_9. *La aplicación del algoritmo iterativo que combina la estimación distribuida con la planificación adaptativa completa la ejecución en 13 iteraciones. El robot* r_1 *realiza la siguiente secuencia de movimientos:* p_8, p_3, p_7, p_3, p_1, p_3, p_1, p_3, p_8, p_6, p_{10}, p_5 *hasta terminar en* p_9. *El robot* r_2 *sigue la trayectoria* p_6, p_{10}, p_5, p_4, p_2, p_{12}, *y llega finalmente a* p_{11}. *La Figura 7 muestra las estimaciones de los dos robots del etiquetado de las regiones en la iteración inicial, en la cuarta y en la última.* ■

7. Conclusiones

En este trabajo se ha abordado la problemática de la planificación de alto nivel en equipos de robots idénticos que han de cumplir una especificación global. Los modelos de redes de Petri considerados permiten obtener modelos escalables con respecto al número de robots. Si se añade un robot más al equipo, la estructura del modelo no cambia. Por otro lado, la planificación se resuelve utilizando problemas de programación matemática, evitándose de este modo la enumeración de los diferentes estados del sistema. En caso de entornos conocidos, se han desarrollado algoritmos que permiten obtener trayectorias de forma automática para los robots y con un coste computacional mucho menor que otras soluciones (Mahulea et al., 2020a).

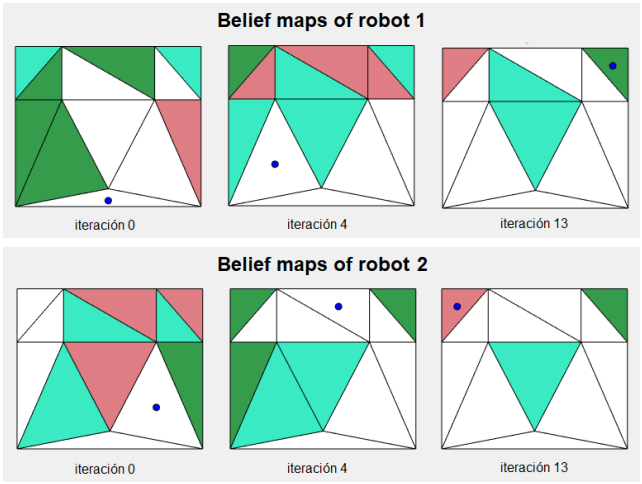

Figura 7: Mapas utilizadas por los robots en diferentes instantes de tiempo para ejecutar el algoritmo de planificación. Cada robot considera que la etiqueta de cada región está dada por la etiqueta de mayor probabilidad. Se observa que en los estados finales (iteración 13), los dos robots tienen una estimación bastante buena de las regiones de interés dadas en la Figura 6.

Son diversas las posibles extensiones de esta línea de investigación. Una primera consiste en considerar la planificación de alto nivel con especificación LTL definida sobre un conjunto de acciones. De esta forma, los robots del equipo se han de coordinar para ejecutar algunas acciones (o tareas), evitando determinadas regiones de interés. Los algoritmos de planificación no tendrán solo que buscar trayectorias para llegar a algunas regiones, también habrán de ejecutar acciones en otras regiones. Para ello, las propiedades inherentes de *localidad* en las transiciones y lugares de las RdPs serán de gran ayuda, al poder *refinar* los modelos de un nivel en otro más detallado.

En el marco de la planificación distribuida se plantean otras vías de trabajo. Por una parte, aunque empíricamente, el método de planificación adaptativa parece funcionar correctamente en diferentes situaciones; no obstante, el acoplamiento existente entre la estimación y el cálculo de las trayectorias hace que la verificación formal de su comportamiento no sea trivial. De cara a poder utilizar estos métodos con garantías, es necesario proveerlos de un marco teórico que especifique las condiciones necesarias para el correcto funcionamiento y terminación, así como el gap de optimalidad con respecto a la solución centralizada con toda la información. También es necesario estudiar con más detalle cómo influye en la planificación la incertidumbre en el marcado de la red; es decir, cómo se comportan los algoritmos en el caso de que los robots no conozcan con exactitud las posiciones del resto del equipo.

La verificación de fórmulas más complejas en este contexto también es un problema abierto en la actualidad. A este respecto, la combinación de los métodos de planificación aquí planteados con técnicas distribuidas de asignación de tareas permite abrir la puerta a la verificación del cumplimiento de especificaciones más avanzadas. Por ejemplo, en casos que se requieran trayectorias repetitivas (infinitas).

Como se ha mencionado, la complejidad computacional del problema de planificación es muy alta. Nuestra intención es la de continuar esta línea de investigación en los siguientes proyectos de investigación. La utilización de la estructura de los modelos basados en las redes de Petri permite reducir la complejidad computacional de los problemas de planificación en sistemas multirobot.

Agradecimientos

Los resultados de esta línea de investigación son fruto de la participación de varios compañeros, investigadores de la Universidad de Zaragoza y de otras Universidades extranjeras. Queremos agradecer la participación de todos ellos, mencionando muy especialmente a Marius Kloetzer (profesor de la Universidad Técnica de Iasi, Rumanía). Este trabajo ha sido financiado parcialmente por los proyectos PGC2018-098719-B-I00 and PGC2018-098817-A-I00 (MCIU/AEI/FEDER, UE) y la ONR Global NICOP grant N62909-19-1-2027.

Referencias

Baier, C., Katoen, J.-P., 2008. Principles of model checking. MIT Press.

Belta, C., Bicchi, A., Egerstedt, M., Frazzoli, E., Klavins, E., Pappas, G.-J., 2007. Symbolic planning and control of robot motion. IEEE Robotics and Automation Magazine 14 (1), 61–71.

Belta, C., Habets, L., 2006. Controlling a class of nonlinear systems on rectangles. IEEE Transactions on Automatic Control 51 (11), 1749–1759.

Brown, F., 2012. Boolean Reasoning: The Logic of Boolean Equations, 2nd Edition. Dover Publications.

Castellanos, J. G., Cervantes, M. V., Santana, J. S., Martínez, S. R., 2014. Seguimiento de trayectorias de un robot móvil (3,0) mediante control acotado. Rev. Iberoamericana de Automática e Informática industrial 11 (4), 426–434.

Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E., Thrun, S., 2005. Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Boston.

Clarke, E.-M.-M., Peled, D., Grumberg, O., 1999. Model checking. MIT Press.

DeCastro, J., Ehlers, R., Runggers, M., Balkan, A., Kress-Gazit, H., 2016. Automated generation of dynamics-based runtime certificates for high-level control. Discrete Event Dynamic Systems 27 (2), 371–405.

Ding, X., Smith, S.-L., Belta, C., Rus, D., 2014. Optimal control of Markov decision processes with linear temporal logic constraints. IEEE Transactions on Automatic Control 59 (5), 1244–1257.

Duret-Lutz, A., Lewkowicz, A., Fauchille, A., Michaud, T., Renault, E., Xu, L., 2016. Spot 2.0 - a framework for ltl and ω-automata manipulation. In: Proc. of ATVA'16. pp. 122–129.

Fainekos, G. E., Girard, A., Kress-Gazit, H., Pappas, G. J., 2009. Temporal logic motion planning for dynamic robots. Automatica 45 (2), 343–352.

Garrido, S., Moreno, L., Gómez, J.-V., Lima, P.-U., 2013. International Journal of Advanced Robotic Systems 10 (1), 64.

Gastin, P., Oddoux, D., 2001. Fast ltl to büchi automata translation. In: Proc. of the 13^{th} Conference on Computer Aided Verification (CAV). pp. 53–65.

González, R., Mahulea, C., Kloetzer, M., 2015. A Matlab-Based Interactive Simulator for Mobile Robotics. In: IEEE CASE'2015: Int. Conf. on Autom. Science and Engineering. Gothenburg, Sweden, pp. 310–315.

González, R., Rodriguez, F., Guzman, J. L., 2014. Autonomous Tracked Robots in Planar Off-Road Conditions. Modelling, Localization and Motion Control. Series: Studies in Systems, Decision and Control. Springer.

Guo, M., Dimarogonas, D.-V., 2015. Multi-agent plan reconfiguration under local LTL specifications. Int. Journal of Robotics Research 34 (2), 218–235.

Habets, L. C. G. J. M., Collins, P. J., van Schuppen, J. H., 2006. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. IEEE Transactions on Automatic Control 51, 938–948.

Julian, B.-J., Angermann, M., Schwager, M., Rus, D., 2012. Distributed robotic sensor networks: An information-theoretic approach. The International Journal of Robotics Research 31 (10), 1134–1154.

Kloetzer, M., Mahulea, C., 2014. A Petri net based approach for multi-robot path planning. Discrete Event Dynamic Systems: Theory and Applications 24 (4), 417–445.

Kloetzer, M., Mahulea, C., 2014. An assembly problem with mobile robots. In: ETFA'2014: IEEE Emerging Technology and Factory Automation. pp. 1–7.

Kloetzer, M., Mahulea, C., 2015. LTL-based planning in environments with probabilistic observations. IEEE Transactions on Automation Science and Engineering 12 (4), 1407–1420.

Kloetzer, M., Mahulea, C., 2020. Path planning for robotic teams based on LTL specifications and Petri net models. Discrete Event Dynamic Systems: Theory and Applications 30 (1), 55–79.

Lacerda, B., Lima, P. U., 2019. Petri net based multi-robot task coordination from temporal logic specifications. Robotics and Autonomous Systems 122, 343–352.

LaValle, S. M., 2006. Planning Algorithms. Cambridge, available at `http://planning.cs.uiuc.edu`.

Leahy, K., Cristofalo, E., Vasile, C.-I., Jones, A., Montijano, E., Schwager, M., Belta, C., 2019. Control in belief space with temporal logic specifications using vision-based localization. The International Journal of Robotics Research 38 (6), 702–722.

Mahulea, C., Kloetzer, M., 2018. Robot Planning based on Boolean Specifications using Petri Net Models. IEEE Trans. on Automatic Control 63 (7), 2218–2225.

Mahulea, C., Kloetzer, M., González, R., 2020a. Path Planning of Cooperative Mobile Robots Using Discrete Event Models. IEEE Wiley.

Mahulea, C., Kloetzer, M., Lesage, J.-J., 2020b. Multi-robot path planning with boolean specifications and collision avoidance. In: WODES'2020: 15th Workshop on Discrete Event Systems.

Mahulea, C., Montijano, E., Kloetzer, M., 2020c. Distributed Multirobot Path Planning in Unknown Maps Using Petri Net Models. IFAC-PapersOnLine21th IFAC World Congress.

Mesbahi, M., Egerstedt, M., 2010. Graph theoretic methods in multiagent networks. Princeton University Press.

Montijano, E., Montijano, J.-I., Sagues, C., Feb 2013. Chebyshev polynomials in distributed consensus applications. IEEE Transactions on Signal Processing 61 (3), 693–706.

Montijano, E., Sagüés, C., 2015. Distributed consensus with visual perception in multi-robot systems. Springer.

Parrilla, L., Mahulea, C., Kloetzer, M., 2017. RMTool: Recent Enhancements. IFAC-PapersOnLine 50 (1), 5824 – 5830, 20th IFAC World Congress.

Schillinger, P., Bürger, M., Dimarogonas, D., 2018. Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. The International Journal of Robotics Research 37 (7), 818–838.

Siegwart, R., Nourbakhsh, I., 2004. Introduction to Autonomous Mobile Robots, First Edition. A Bradford book. The MIT Press, USA.

Silva, M., 1985. Las Redes de Petri : en la Automática y la Informática; 1^{a} ed. Editorial AC Madrid.

Silva, M., Colom, J.-M., 1988. On the Computation of Structural Synchronic Invariants in P/T Nets. Advances in Petri Nets'87 340, 386–417.

Silva, M., Teruel, E., Colom, J.-M., 1998. Linear Algebraic and Linear Programming Techniques for the Analysis of P/T Net Systems. Lecture on Petri Nets I: Basic Models 1491, 309–373.

Tumova, J., Dimarogonas, D., 2016. Multi-agent planning under local LTL specifications and event-based synchronization. Automatica 70, 239–248.

Ulusoy, A., Smith, S., Ding, X., Belta, C., 2012. Robust multi-robot optimal path planning with temporal logic constraints. In: ICRA 2012: IEEE Conference on Robotics and Automation. pp. 4693–4698.

Wolper, P., Vardi, M., Sistla, A., 1983. Reasoning about infinite computation paths. In: Proc. of the 24th IEEE Symposium on Foundations of Computer Science. pp. 185–194.

Yen, J.-Y., 1971. Finding the k shortest loopless paths in a network. Management Science 17 (11), 712–716.