USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
ESCOLA POLITÉCNICA SUPERIOR DE ENXEÑARÍA

Memoria práctica 3
Agentes Inteligentes

Losada Álvarez Adrián
Seoane Temprano Pablo

TABLA DE CONTENIDO

1. INTRODUCCIÓN

En el ámbito de la robótica y la planificación de acciones autónomas, el desarrollo de sistemas capaces de tomar decisiones inteligentes y realizar tareas complejas es fundamental. La planificación automática se erige como una herramienta esencial para diseñar algoritmos y estrategias que permitan a los robots realizar acciones de manera autónoma y eficiente.

En el marco de este contexto, esta práctica se enfoca en la aplicación de la planificación automática y la modelización de problemas utilizando lógica de predicados, haciendo uso del paquete ROSPlan de ROS. A lo largo de esta práctica, se exploran y desarrollan una serie de ejercicios progresivos que abarcan desde el movimiento básico de un robot en un entorno discreto hasta la planificación multi-agente centralizada.

El documento proporciona una guía detallada que comprende desde la instalación inicial de los componentes necesarios hasta la implementación y ejecución de acciones complejas, todo ello en un entorno simulado. Se abordan ejercicios que van desde el desplazamiento y giro del robot hasta la manipulación de objetos y la optimización de tareas considerando costos y recursos.

A través de la comprensión y aplicación de los conceptos de planificación, modelización de problemas y ejecución de planes, esta práctica proporciona una plataforma sólida para el aprendizaje y la experimentación en el campo de la planificación autónoma en entornos robóticos simulados. Este informe detalla cada etapa del proceso, desde la conceptualización teórica hasta la implementación práctica, ofreciendo una visión integral de la aplicación de la planificación automática en la robótica.

2. EJERCICIOS

EJERCICIO 1

En comparación con las plantillas originales *'domain.pddl'* y *'problem.pddl'*, las modificaciones realizadas en *'domain1.pddl'* y *'problem1.pddl'* para el ejercicio 1 se enfocan en:

Cambios en *'domain1.pddl'* respecto a *'domain.pddl'*:

Definición de Tipos:

- Se agregaron tipos específicos: *'waypoint'*, *'robot'*, y *'orientation'*.

Predicados:

- Se incluyeron predicados adicionales:
  (robot_at ?r - robot ?wp1 - waypoint ?o - orientation): Indica la posición y orientación del robot en un waypoint dado.
  (connected ?wp1 ?wp2 - waypoint ?o - orientation): Define la conexión entre dos waypoints en una orientación específica.

Acción 'move':

- Se creó la acción 'move':
  Parámetros: '?r - robot', '?wp1 - waypoint', '?wp2 - waypoint', '?o - orientation'.
  Precondición: El robot debe estar en un *waypoint* específico y debe existir una conexión entre ese *waypoint* y el siguiente en la orientación dada.
  Efecto: Mueve al robot desde un *waypoint* a otro en la orientación indicada.

Cambios en *'problem1.pddl'* respecto a *'problem.pddl'*:

Definición de Objetos:

- Se definieron objetos específicos: *'wp00'*, *'wp01'*, ..., *'wp33'* como waypoints; *'down'*, *'up'*, 'left', *'right'* como orientaciones; y *'kenny'* como el robot.

Inicialización:

- Se incluyeron predicados de inicio para:
- Posicionar al robot *'kenny'* en el *waypoint 'wp00'* con orientación *'down'*.
- Establecer las conexiones entre waypoints: *'wp00'* con *'wp10'*, *'wp10'* con *'wp20'*, y *'wp20'* con *'wp30'*, todas en orientación *'down'*.

Meta:

- Se fijó la meta de que el robot *'kenny'* debe estar en el *waypoint 'wp30'* con orientación *'down'*.

Estas modificaciones en *'domain1.pddl'* y *'problem1.pddl'* adaptan los archivos para permitir el movimiento del robot entre waypoints con restricciones específicas de conexión y posición.

Además, se proporciona un video demostrativo del comportamiento: Ejercicio_1

EJERCICIO 2

Cambios en *'domain2.pddl'* respecto a *'domain.pddl'*:

Acción 'turn':

- Se agregó la acción *'turn'*:
  Parámetros: '?r - robot', '?wp - waypoint', '?o1 - orientation' y '?o2 - orientation'.
  Precondición: Requiere que el robot esté en un waypoint específico con una orientación inicial.
  Efecto: Cambia la orientación del robot en el mismo *waypoint*.

Cambios en *'problem2.pddl'* respecto a *'problem.pddl'*:

Inicialización:

- Se mantuvieron los objetos, orientaciones y el robot definido.
- Se mantuvo la posición inicial del robot *'kenny'* en *'wp00'* con orientación *'down'*.
- Se añadió una conexión adicional entre *'wp30'* y *'wp31'* en la orientación *'right'*.

Meta:

- La meta será que el robot *'kenny'* esté en *'wp31'* con orientación *'right'*.

En este caso, se introdujo la acción *'turn'* en el dominio para permitir al robot cambiar su orientación en un mismo *waypoint*, y en el problema se estableció una nueva conexión entre dos waypoints para reflejar un cambio en la topología del entorno.

Además, se proporciona un video demostrativo del comportamiento: Ejercicio_2

EJERCICIO 3

Cambios en *'domain3.pddl'* respecto a *'domain.pddl'*:

Acción 'push':

- Se agregó la acción *'push'*:
  Parámetros: '?r - robot', '?b - box', '?wpr - waypoint', '?wpb1 - waypoint', '?wpb2 - waypoint', '?o - orientation'.
  Precondición: Requiere que el robot esté en un *waypoint*, haya una caja en un *waypoint* adyacente vacío y esté conectado con otro *waypoint* vacío.
  Efecto: Mueve la caja a otro *waypoint* adyacente vacío, actualizando las posiciones del robot y la caja.

Predicados 'empty' y 'box_at':

- Se añadieron los predicados *'empty'* y *'box_at'* para rastrear los waypoints vacíos y las ubicaciones de las cajas respectivamente.

Cambios en *'problem3.pddl'* respecto a *'problem.pddl'*:

Inicialización:

- Se incluyeron los objetos *'green_box'* y *'blue_box'* como cajas en el entorno.
- Se añadieron predicados *'box_at'* para definir las ubicaciones iniciales de las cajas *'green_box'* y *'blue_box'*.
- Se establecieron los waypoints *'wp13'* y *'wp21'* como vacíos con el predicado *'empty'*.
- Se definió una nueva topología agregando conexiones entre waypoints para reflejar los movimientos posibles del robot y las ubicaciones de las cajas.

Meta:

- La meta incluye la posición final del robot *'kenny'* en *'wp12'* con orientación *'right'*, y las cajas *'green_box'* y *'blue_box'* en *'wp21'* y *'wp13'* respectivamente.

Estos cambios permiten al robot mover cajas a waypoints adyacentes vacíos, ampliando así las capacidades de planificación para manejar las cajas en el entorno.

Además, se proporciona un video demostrativo del comportamiento: Ejercicio_3

EJERCICIO 4

Cambios en *'domain4.pddl'* respecto a *'domain.pddl'*:

Función 'battery-level':

- Se añadió la función *'battery-level'* para rastrear el nivel de batería del robot.

Acciones con costo de batería:

- Se modificaron las acciones *'move'* y *'push'* para incorporar un costo de batería:
  1- Se incluyeron precondiciones que requieren un nivel mínimo de batería para ejecutar las acciones.
  2- Se agregaron efectos para disminuir el nivel de batería después de completar las acciones. *'move'* reduce la batería en 1 unidad, mientras que *'push'* la reduce en 2 unidades.

Cambios en *'problem4.pddl'* respecto a *'problem.pddl'*:

Función de batería y métrica:

- Se definió el nivel inicial de la batería del robot como '10' con el predicado *'(= (battery-level) 10)'*.
- Se incluyó la métrica *'maximize (battery-level)'* para maximizar la cantidad de batería restante como objetivo.

Meta:

- La meta seguirá siendo la misma.

Estos cambios permiten modelar un robot con una función de batería que afecta el costo de las acciones *'move'* y *'push'*, añadiendo una dimensión de optimización para maximizar la batería restante al finalizar el plan.

Además, se proporciona un video demostrativo del comportamiento: Ejercicio_4

EJERCICIO 5

Aunque simplificado en esta práctica, la base de hechos del robot debería actualizarse continuamente a partir de la información sensorial. Un cambio en el entorno actualizará la base de hechos, lo que podría causar la falla del plan y requerir que el robot replanifique en busca de un nuevo plan. En este ejercicio, simularemos uno de estos eventos.

Para ello, vamos a hacer uso del script *'updateKB.bash'* del directorio scripts del paquete *'ai_planning'*. Este script provoca la eliminación de un predicado de la base de hechos, en particular, elimina la conexión entre dos waypoints. Este ejemplo elimina el predicado *'connected'* que une *'wp00'* con *'wp01'* en la orientación right. Se debe modificar el nombre *'connected'* con el nombre del predicado que se haya elegido para unir dos *waypoints*, así como *'wp00'*, *'wp01'* y *'right'* con el nombre de los *waypoints* y su orientación correspondiente.

Durante la ejecución del plan en Gazebo, se elimina una conexión que el robot va a utilizar más adelante en el plan. ¿Qué sucede en este punto? ¿Cuál es el contenido de la base de hechos en este momento? ¿Qué se te ocurre que podrías hacer para que el robot continúe con la ejecución?
El plan que tenía el robot falla debido a que la conexión de las casillas por las que tenía que pasar desaparece, por lo que para poder resolver el problema busca otra ruta alternativa para llegar a la misma casilla a la que se dirigía.

Cambios en *'domain5.pddl'* respecto a *'domain.pddl'*:

Para este ejercicio no fue necesario modificar nada en la definición del dominio.

Cambios en *'problem4.pddl'* respecto a *'problem.pddl'*:

Función de batería y métrica:

- Se definieron nuevas conexiones y rutas entre waypoints para el escenario.
- Se modificaron las conexiones de waypoints para simular rutas alternativas.
- Se agregaron conexiones adicionales para mover las cajas a sus ubicaciones finales.
- Se mantuvo la métrica 'maximize (battery-level)' para maximizar la cantidad de batería restante como objetivo.

Meta:

- La meta seguirá siendo la misma.

EJERCICIO 6

Cambios en *'domain6.pddl'* respecto a *'domain.pddl'*:

Acciones durativas agregadas:

- **'move_robot'**: Permite al robot moverse entre waypoints.
- **'turn_robot'**: Facilita el giro del robot hacia una nueva orientación.
- **'push_box'**: Permite al robot empujar una caja hacia un *waypoint* adyacente.

Precondiciones y efectos de las acciones durativas:

- Cada acción tiene sus propias precondiciones y efectos relacionados con las ubicaciones del robot, cajas y waypoints.

Predicados y tipos extendidos:

- Se han agregado predicados relacionados con las acciones durativas y las nuevas condiciones que involucran tiempo y movimiento.

Cambios en *'problem6.pddl'* respecto a *'problem.pddl'*:

Inclusión de robots y cajas:

- Se han añadido objetos *'kenny1'*, *'kenny2'*, *'green_box'*, y *'blue_box'* como elementos en el entorno.
- Los robots y cajas se inicializan en ubicaciones específicas.

Definición de condiciones iniciales:

- Se establecen conexiones entre waypoints y se inicializan ciertos waypoints como vacíos.

Meta:

- Se define la condición final deseada, que involucra la posición final de los robots y la ubicación final de las cajas.

3. CONCLUSIÓN

A lo largo de esta práctica, hemos explorado varios ejercicios relacionados con la planificación en entornos robóticos simulados. Desde la creación de dominios hasta la definición de problemas específicos, cada ejercicio ha presentado desafíos y mejoras progresivas en la complejidad de la planificación.

Progresión de la complejidad: Cada ejercicio presentó nuevos elementos como tipos, predicados, acciones, y cambios en la topología del entorno. Esto nos llevó desde tareas básicas de movimiento hasta la manipulación de cajas y la consideración del estado de la batería del robot.

Uso de diferentes acciones y predicados: Desde acciones simples como mover y girar hasta acciones más complejas como empujar cajas, cada nuevo ejercicio agregó funcionalidades y capacidades más avanzadas al robot.

Importancia de la actualización de la base de hechos: En el ejercicio 5, se exploró la necesidad de replanificar ante cambios en el entorno que afectan la base de hechos del robot, demostrando la importancia de adaptarse a información sensorial actualizada.

Acciones durativas en el último ejercicio: El ejercicio 6 introdujo acciones durativas, considerando la planificación en un marco temporal y permitiendo al robot tomar decisiones secuenciales a lo largo del tiempo.

En conjunto, estos ejercicios nos han mostrado la evolución y complejidad creciente de la planificación en entornos robóticos simulados. Desde tareas básicas hasta la consideración de tiempo y recursos, estos ejercicios han proporcionado una visión amplia de cómo la planificación se vuelve más sofisticada para abordar desafíos robóticos cada vez más complejos.