Tema 2. Planificación inteligente. Práctica 1

Consiste en la implementación en pyhop de un problema para planificar las tareas a realizar por un rover en Marte. Estas tareas son tres: (1) recoger, analizar y enviar los resultados sobre muestras de suelo, (2) recoger, analizar y enviar los resultados sobre muestras de roca y (3) tomar imágenes y enviarlas.

Un rover puede estar equipado con instrumental para tomar muestras de suelo, para tomar muestras de roca y/o para tomar imágenes en diferentes modos (color, alta resolución, baja resolución).

```
state1.equipped_for_soil_analysis={'r0'}
state1.equipped_for_rock_analysis={'r0'}
state1.equipped_for_imaging={'r0'}
state1.supports={'r0':{'c0':{'colour','high-res'}}}
state1.on_board={'r0':{'c0'}}
```

Cada cámara puede tomar imágenes de diferentes objetivos y debe estar calibrada para ello.

```
state1.calibration_target={'c0':'o1'}
state1.calibrated={'r0':{'c0':False}}
```

Un rover tiene un compartimento de capacidad 1 donde puede almacenar las muestras que recoge.

```
state1.store_of={'r0':'r0store'}
state1.empty={'r0store':True}
```

Se conocen los puntos donde pueden recogerse muestras de cada tipo y desde qué puntos (waypoints) se pueden tomar imágenes de un determinado objetivo.

```
state1.at_soil_sample={'w0','w2','w3'}
state1.at_rock_sample={'w1','w2','w3'}
state1.visible_from={'o1':{'w0','w1','w2','w3'}}
```

Asimismo, se conoce los “caminos” que puede atravesar el rover para moverse de un punto a otro y los puntos que son visibles desde cada punto.

```
state1.can_traverse={'r0':[['w0','w3'],['w1','w3'],['w1','w2'],['w2','
w0'],['w3','w0'],['w3','w1'],['w2','w1'],['w0','w2']]}
state1.visible={'w0':{'w1','w2','w3'}, 'w1':{'w0','w2','w3'},
'w2':{'w0','w1','w3'}, 'w3':{'w0','w1','w2'}}
```

El rover debe comunicar sus análisis a un Lander. Se conoce la posición de ambos.

```
state1.at_lander='w0'
state1.at={'r0':'w3'}
```

Además, el estado almacena información auxiliar necesaria para el proceso de planificación. Así, se almacena qué muestras ha recogido un rover, qué análisis ha comunicado y qué puntos ha recorrido.

```
state1.have_rock_analysis={'r0':[]}
state1.have_soil_analysis={'r0':[]}
state1.have_image={'r0':[]}
state1.communicated_soil_data=[]
state1.communicated_rock_data=[]
state1.communicated_image_data=[]
state1.crossed={'r0':[]}
```

El objetivo se especifica como una muestra a tomar, indicando el punto donde debe tomarse, de qué tipo es y si es una imagen, el modo.

```
pyhop.pyhop(state1, [('communicate_soil_data','w2')], verbose=3)
pyhop.pyhop(state1, [('communicate_rock_data','w3')], verbose=3)
pyhop.pyhop(state1, [('communicate_image_data','o1','high-res')],
verbose=3)
```

Con el estado inicial y los objetivos proporcionados en el fichero “rovers_problem”, con algunos operadores ya implementados en el fichero “rovers_operators” y con un método implementado en el fichero “rovers_methods”, se pide implementar el conocimiento experto que se describe a continuación:

Operadores:

1. Already_there (se proporciona): Este operador recibe el estado y el rover e inicializa el atributo crossed con la posición actual del rover. Crossed se utiliza como base para la tarea Navigate y para el método Go_to_waypoint.
2. Navigate (se proporciona): Este operador recibe el estado, el rover y el destino y si el destino es visible desde la posición actual, actualiza la posición actual con el destino.
3. Sample_soil: Este operador recibe el estado y el rover y comprueba si éste se encuentra en una posición en la que puede recoger una muestra de suelo y si su compartimento está vacío. Si es así, se recoge la muestra, lo que implica actualizar el estado del compartimento (ahora está lleno), indicar que tiene la muestra recogida en have_soil_analysis y eliminar la muestra recogida de at_soil_sample.
4. Sample_rock: Operador similar al anterior pero para recoger una muestra de roca.
5. Drop: Este operador recibe el estado actual y el rover y vacía el compartimento de este rover, si estaba lleno.
6. Calibrate: Este operador recibe el estado, el rover, la cámara a utilizar y el objetivo del cual se va a tomar la imagen. Si dicho objetivo se encuentra en la lista de los que la cámara puede tomar, el objetivo es visible desde la posición actual del rover y la cámara no está calibrada, ésta pasa a estar calibrada.
7. Take_image: Este operador recibe estado, el rover, la cámara a utilizar, el objetivo del cual se va a tomar la imagen y el modo de dicha imagen. Si la cámara está calibrada, se toma la imagen (se actualiza have_image) y la cámara ya no se encuentra calibrada.
8. Communicate_soil_data_to_lander: Este operador recibe el estado, el rover y la muestra (posición) tomada. Si el rover ha recogido dicha muestra previamente y la posición del Lander es visible desde la posición actual del rover, se envía la información (lo que implica actualizar communicated_soil_data)

9. Communicate_rock_data_to_lander: Este operador a la anterior, pero con una muestra de roca
10. Communicate_image_data_to_lander: Este operador a la anterior, pero con una imagen, por tanto se recibe, además del estado y del rover, el objetivo y el modo de imagen.

Métodos:

1. Go_to_waypoint_m (se proporciona): Este método recibe el estado, el rover y el waypoint destino. Se comprueba si ya se encuentra en dicho destino, en ese caso, el método se descompone en el operador Already_there. En otro caso, se busca el siguiente waypoint a visitar y se descompone en el operador Navigate y la tarea Go_to_waypoint.
2. Communicate_soil_data_m: Este método recibe el estado y el waypoint donde debe tomarse la muestra de suelo. Comprueba si el rover está equipado para tomar muestras de suelo y si es así, se descompone en la tarea Go_to_waypoint y los operadores Sample_soil, Drop y Communicate_soil_data_to_lander.
3. Communicate_rock_data_m: Este método recibe el estado y el waypoint donde debe tomarse la muestra de roca. Comprueba si el rover está equipado para tomar muestras de roca y si es así, se descompone en la tarea Go_to_waypoint y los operadores Sample_rock, Drop y Communicate_rock_data_to_lander.
4. Communicate_image_data_m: Este método recibe el estado, el objetivo del cual se va a tomar la imagen y el modo. Comprueba si el rover está equipado para tomar imágenes y recorre las distintas cámaras con las que puede estar equipado para encontrar aquella que soporta el modo requerido. Se descompone en los operadores Calibrate, Take_image y Communicate_image_data_to_lander.

Tareas:

Cada uno de los métodos especificados anteriormente corresponde con una tarea, tal y como se muestra en el fichero “rovers_methods”.

NOTA:

Se sugiere implementar en primer lugar los siguientes operadores: Sample_soil, Drop, Communicate_soil_data_to_lander y el método Communicate_soil_data_m, con el fin de ejecutar el planificador con un tipo de muestra.