Práctica 3
Técnica sistemas inteligentes

UNIVERSIDAD DE GRANADA

Antonio Manuel Fresneda Rodríguez

antoniomfr@correo.ugr.es

Ejercicio 0

Este ejercicio comprobamos que el dominio y el problema funcionan correctamente. La salida del planificador esta en el archivo EJ0.txt

Ejercicio 1

Este problema no se puede resolver. Esto ocurre debido a que la tarea de transportar a una persona que no esté en el mismo lugar que el avión no está definida. Para resolverlo la hemos implementado he definido un nuevo caso para la tarea.

- *Precondiciones*: Que el avión esté en cA, la persona esté en cP y cp sea distinto de cA.
- *Acciones*: El avión va de Ca a Cp, la persona embarca en el avión y el avión vuela a la ciudad de destino del pasajero.

La salida del plan esta en el archivo EJ1.txt en la carpeta EJ1.

Ejercicio2

En este problema nos dicen que tenemos que tener en cuenta el combustible.
El dominio del ejercicio anterior no tenía en cuenta que nos quedásemos sin combustible. Para ello he añadido *method: fuel-no-suficiente* a la tarea: *mover-avion*.

- *Precondiciones*: Que no haya fuel para viajar de la ciudad c1 a c2.
- *Acciones*: Recargamos el depósito en la ciudad 1 y volamos después a la ciudad 2.

La salida del plan está en el archivo EJ2.txt en la carpeta E2.

Ejercicio3

Para poder representar en nuestro dominio que el avión puede tener dos tipos de velocidad (lento y rápida) y dos tipo de consumo (lento y rápido) vamos a tener que modificar los predicados derivados para estimar si vamos a tener fuel y la tarea de moverse del avión, ya que va a recorrer más distancia en menos tiempo.
He declarado la función *(fuel-limit)* y he añadido los predicados *(hay-fuel-viaje-lento ?a ?c1 ?c2)* y *(hay-fuel-viaje-rapido ?a ?c1 ?c2)*
El derivado de cada uno deduce que el el fuel del avión ?a sea mayor que el consumo volado de forma lenta/rápida multiplicado por la distancia entre las

ciudades ?c1 y ?c2.
Una vez hecho esto, he modificado la tarea de mover el avión añadiendo lo siguiente:

- *fuel-suficiente-viaje-(lento o rápido)*
  - *Precondiciones*: Que el avión tenga suficiente fuel para viajar de ?c1 a ?c2 viajando de forma lenta/rápida y que el total de fuel usado (que se incrementa cada vez que se vuela de un sitio a otro) sea menor que el limite de fuel.
  - *Tarea*: Volar de ?c1 a ?c2
- *fuel-no-suficiente-viaje-(lento o rápido)*
  - *Precondiciones*: Que el avión no tenga suficiente fuel para viajar de ?c1 a ?c2 viajando de forma lenta/rápida y que el total de fuel usado (que se incrementa cada vez que se vuela de un sitio a otro) sea menor que el limite de fuel.
  - *Tarea*:Repostar el avión en ?c1 y Volar de ?c1 a ?c2

La salida del plan está en el archivo EJ3.txt

Ejercicio 4

En este ejercicio se nos propone varios apartados en los que se pide que modifiquemos las acciones *board* y *debark*.

Apartado a

Para representar la capacidad máxima de un avión he usado la función llamada *capacity ?a* que representa el número máximo de pasajeros que pueden ir en el avión a y otra función llamada *pasajeros a* que representa el numero de pasajeros que hay en el avión a (inicialmente 0).
Una vez hecho esto se ha modificado el la tarea de embarque y desembarque para que incremente/disminuya el contador.

Apartado b

Como se indica en el enunciado se ha añadido el predicado *destino ?x - person ?y - city* para que el planificador tenga información sobre el destino de la persona x.
Para representar que se puedan embarcar varias personas en la misma ciudad he hecho lo siguiente:

- Las acciones board y debark se han cambiado el nombre a board-passanger y debark-passanger puesto que van a representar que un **único** pasajero va a entrar en el avión.

- Se han añadido dos nuevas tareas:
  - *embarcar*: Esta es una tarea que recursivamente llama a la accion *board-passanger*.
    El primer *method* se llama *embarca-pasajero*
    - *Precondiciones*: Que el avión y la persona que va a embarcar estén en la ciudad c, que no se haya superado el máximo de pasajeros en el avión, y que el destino no sea la ciudad c.[1]
    - *Acciones*: Embarcar pasajero y la llamada recursiva.

    El segundo *method* es la condición de parada de la recurrencia.
  - *desembarcar*: Esta es una tarea que recursivamente llama a la accion *debark-passanger*.
    El primer *method* se llama *desembarca-pasajero*
    - *Precondiciones*: Que el avión y la persona que va a embarcar estén en la ciudad c.
    - *Acciones*: Embarcar pasajero y la llamada recursiva.

    El segundo *method* es la condición de parada de la recurrencia.

No he realizado el apartado 3 de este ejercicio.

Experimentación

He planteado 4 experimentos:

- En el ejemplo 1 he considerado 6 personas y un avión distribuidos: p1 Almeria, p2 Almeria, p3 Madrid, p4 Madrid, p5 Granada, p6 Granada, a1 Madrid.
  Los destinos son: p1 Gibraltar, p2 Jaen, p3 Granada, p4 Bilbao, p5 Jaen, p6 Gibraltar.
- En el ejemplo 2 he considerado 20 personas y dos aviones que estan distribuidos: p1 Almeria, p2 Almeria, p3 Sevilla, p4 Sevilla, p5 Madrid, p6 Madrid, p7 Granada, p8 Granada, p9 Bilbao, p10 Bilbao, p11 Almeria, p12 Almeria, p13 Sevilla, p14 Sevilla, p15 Madrid, p16 Madrid, p17 Granada, p18 Granada, p19 Bilbao, p20 Bilbao, a1 Madrid, a2 Madrid.
  Los aviones tienen distinto fuel (15000/10000), la misma velocidad y consumo en lento, a2 tiene el doble de velocidad y quema el doble de fuel que a1 cuando va rápido. Los destinos son los siguientes: p1 Gibraltar, p2

[1] Si no ponemos esta precondición se puede producir un bucle en la recurrencia, ya que cuando los pasajeros desembarcan en una ciudad y hay no se supera el limite, se cumplen las precondiciones de *embarca-pasajero*

Almeria, p3 Madrid, p4 Gibraltar, p5 Gibraltar, p6 Granada, p7 Gibraltar, p8 Gibraltar, p9 Granada, p10 Bilbao, p11 Madrid, p12 Madrid, p13 Madrid, p14 Gibraltar, p15 Gibraltar, p16 Granada, p17 Gibraltar, p18 Gibraltar, p19 Granada, p20 Bilbao.

Las salidas de los planes están en los archivos EJ4_numero_del_ejemplo.txt