



UNIVERSIDAD DE GRANADA

GRADO INGENIERÍA INFORMÁTICA (2016 – 2017)

TÉCNICAS SISTEMAS INTELIGENTES

Práctica 3 | Planificador HTN

Trabajo realizado por: Antonio Miguel Morillo Chica

1. Introducción.

En esta practica ahora vamos a usar el planificador htnp creado por el departamento de decsai. Es un planificaco como ff-metric basado en el uso de reglas y con sintaxis pddl. Además tiene un debugger a diferencia de ff-metric.

2. Ejercicios realizados.

A continuación explicaré como he resuelto los distintos problemas expuestos para la segunda entrega de la practica de planificación.

- **Ejercicio 1:** *Consiste en transportar 3 personas (inicialmente en las ciudades C1, C2 y C3) a la ciudad C5, considerando que el avión está en la ciudad C4. Se asume al igual que en el problema ejemplo que no hay restricciones de fuel. Comprobar que con el dominio entregado como ejemplo HTNP no encuentra solución y modificar el dominio para que la encuentre.*

Para resolver el problema es necesario tener en cuenta que la persona y el avión podrían no encontrarse en la misma ciudad por lo que habría que desplazar a el avión a la ubicación de la persona.

Esto se resuelve muy fácilmente, lo único que hay que modificar el task de trasportar persona añadiendo un nuevo method en el que se contemplan lo mencionado en el párrafo de arriba.

```
(:method Case3
  :precondition (and
    (at ?p - person ?c1 - city)
    (at ?a - aircraft ?c2 - city)
    (different ?c1 - city ?c2 - city)
  )
  :task (
    (fly ?a ?c2 ?c1)
    (board ?p ?a ?c1 ?c)
    (mover-avion ?a ?c1 ?c)
    (debark ?p ?a ?c)
  )
)
```

```
Símbolo del sistema
C:\Users\miguel>cd Desktop\Entrega2\htnp
C:\Users\miguel\Desktop\Entrega2\htnp>htnp -d zenotravel-v01.pddl -p problema-zeno-v01.pddl

zenotravel-v01.pddl:29: warning: Duplicated variable ? ? `?x'.
#####
0 errors 1 warnings parsing domain file: zenotravel-v01.pddl
-----
:action (fly a1 c4 c1) start: 05/06/2007 08:00:00 end: 05/06/2007 23:00:00
:action (board p1 a1 c1) start: 05/06/2007 23:00:00 end: 06/06/2007 00:00:00
:action (fly a1 c1 c5) start: 06/06/2007 00:00:00 end: 06/06/2007 10:00:00
:action (debark p1 a1 c5) start: 06/06/2007 10:00:00 end: 06/06/2007 11:00:00
:action (fly a1 c5 c2) start: 06/06/2007 11:00:00 end: 07/06/2007 02:00:00
:action (board p2 a1 c2) start: 07/06/2007 02:00:00 end: 07/06/2007 03:00:00
:action (fly a1 c2 c5) start: 07/06/2007 03:00:00 end: 07/06/2007 18:00:00
:action (debark p2 a1 c5) start: 07/06/2007 18:00:00 end: 07/06/2007 19:00:00
:action (fly a1 c5 c3) start: 07/06/2007 19:00:00 end: 08/06/2007 10:00:00
:action (board p3 a1 c3) start: 08/06/2007 10:00:00 end: 08/06/2007 11:00:00
:action (fly a1 c3 c5) start: 08/06/2007 11:00:00 end: 09/06/2007 02:00:00
:action (debark p3 a1 c5) start: 09/06/2007 02:00:00 end: 09/06/2007 03:00:00
Number of actions: 12 (12)
Expansions: 6
Generated nodes: 18
Inferences: 0
Time in seconds: 0
Real Time: -1.17188e-05
Used Time: 1.01328e-09
System Time: -1.60933e-09
C:\Users\miguel\Desktop\Entrega2\htnp>
```

- **Ejercicio 2:** *Consiste en asumir que hay restricciones de fuel. El fuel inicial del avión es de 200 y la capacidad total de 300. Deben contemplarse ahora acciones de repostaje. La situación de partida de personas y avión es la misma que en el problema anterior.*

Ahora además tendremos que tener en cuenta el fuel necesario para realizar todo el camino, para ello hay que tener en cuenta que el avión debe repostar si ve que se va a quedar sin fuel para poder seguir realizando

Rescato el domino anterior un método, en mi caso, fuel-insuficiente. Una vez repostado vuela a la ciudad que debe:

```
(:metod fuel-insuficiente
  :precondition (not (hay-fuel ?a ?c1 ?c2))
  :tasks (
    (refuel ?a ?c1)
    (fly ?a ?c1 ?c2)
  )
)
```

Si no hay fuel suficiente lo que hay que hacer es, claramente, repostar.

```

Simbolo del sistema
zenotravel-v02.pddl:29: warning: Duplicated variable ?x'.
#####
0 errors 1 warnings parsing domain file: zenotravel-v02.pddl
-----
:action (fly a1 c4 c1) start: 05/06/2007 08:00:00 end: 05/06/2007 23:00:00
:action (board p1 a1 c1) start: 05/06/2007 23:00:00 end: 06/06/2007 00:00:00
:action (refuel a1 c1) start: 06/06/2007 00:00:00 end: 16/06/2007 10:00:00
:action (fly a1 c1 c5) start: 16/06/2007 10:00:00 end: 16/06/2007 20:00:00
:action (debark p1 a1 c5) start: 16/06/2007 20:00:00 end: 16/06/2007 21:00:00
:action (fly a1 c5 c2) start: 16/06/2007 21:00:00 end: 17/06/2007 12:00:00
:action (board p2 a1 c2) start: 17/06/2007 12:00:00 end: 17/06/2007 13:00:00
:action (refuel a1 c2) start: 17/06/2007 13:00:00 end: 27/06/2007 23:00:00
:action (fly a1 c2 c5) start: 27/06/2007 23:00:00 end: 28/06/2007 14:00:00
:action (debark p2 a1 c5) start: 28/06/2007 14:00:00 end: 28/06/2007 15:00:00
:action (fly a1 c5 c3) start: 28/06/2007 15:00:00 end: 29/06/2007 06:00:00
:action (board p3 a1 c3) start: 29/06/2007 06:00:00 end: 29/06/2007 07:00:00
:action (refuel a1 c3) start: 29/06/2007 07:00:00 end: 11/07/2007 19:00:00
:action (fly a1 c3 c5) start: 11/07/2007 19:00:00 end: 12/07/2007 10:00:00
:action (debark p3 a1 c5) start: 12/07/2007 10:00:00 end: 12/07/2007 11:00:00
Number of actions: 15 (15)
Expansions: 6
Generated nodes: 21
Inferences: 0
Time in seconds: 0
Real Time: -5.85938e-06
Used Time: -8.9407e-11
System Time: 3.35276e-10
C:\Users\miguel\Desktop\Entrega2\http>

```

- **Ejercicio 3:** *Consiste en considerar acciones de vuelo lento y rápido para tratar de transportar las personas lo más rápido posible con un límite de fuel. En el dominio se tienen que codificar los métodos y tareas de manera que se priorice el uso de acciones de velocidad rápida. El límite de fuel se define con la función (fuel-limit) y es asignado en el estado inicial a 1500. La suma total de fuel gastado en todos los transportes no puede superar 1500 unidades, pero el avión debe viajar siempre lo más rápido posible.*

Lo primero he hecho ha sido crear el predicado fuel-limit. He modificado el hay fuel de la siguiente forma para que la distancia con combustible para que tenga en cuenta la distancia posible según el fuel:

```
(> (fuel ?a) (*(distance ?c1 ?c2)(slow-burn ?a)))
```

Tras esto he querido crear métodos para controlar el uso del zoom pero no he podido conseguirlo, aunque he implementado los siguientes:

```

(:derived
  (hay-fuel-rapido ?a - aircraft ?c1 - city ?c2 - city)
  (> (fuel ?a) (*(distance ?c1 ?c2) (slow-burn ?a)))
)

(:method fuel-suficiente-rapido
  :precondition (hay-fuel-rapido ?a ?c1 ?c2)

```

```
) :tasks ( (zoom ?a ?c1 ?c2 ) )
```