Técnicas de los sitemas inteligentes
Práctica 3

Guillermo Gómez Trenado | 77820354-S

4 de junio de 2018

Ejercicio 1. Modificar el problema para recoger a viajeros en otras ciudades

Para solucionar este problema sólo tenemos que añadir un método nuevo a **transport-person** que comprueba si el pasajero está en otra ciudad, va a ella y llama recursivamente a transport-person.

Ejercicio 2. Restricciones de fuel

En primer lugar, aquí observé que los predicados derivados **igual** y **diferente** no funcionaban, así que para solucionarlo añadimos en *requirements* **:equality** y ya podemos comparar la igualdad como *(= (a b))*.

Ahora sí, para resolver el problema he añadido dos funciones derivadas, **hay-fuel** y **llega**.

La primera comprueba si podría llegar al destino con el combustible actual y el segundo si es técnicamente posible si repostara. Ahora separamos **mover-avion** en dos métodos distintos para la situación de suficiente combustible o no suficiente.

Si tiene suficiente combustible va al destino, si no lo tiene reposta y vuelve a llamarse recursivamente.

Ejercicio 3. Dos velocidades de vuelo

Para resolver este ejercicio primero utilicé las constantes *slow* y *fast* para conservar los métodos *llega* y *hay-fuel* con un parámetro más que fuera la velocidad, sin embargo dio problemas con *:equality* y separé las dos funciones derivadas *hay-fuel* y *llega* en cuatro atendiendo a la velocidad.

Del mismo modo, separé los dos métodos en *mover-avion* en cuatro nuevos métodos **fuel-suficiente-fast**, **fuel-insuficiente-fast**, **fuel-suficiente-slow** y **fuel-insuficiente-slow**, funcionan igual que en el caso anterior pero llamando a *zoom* o *fly* respectivamente.

Ejercicio 4. Problema completo

1. *Acciones board y debark*

   Para este paso sólo añadí la nueva función *(board-limit ?a -aircraft)* que se define en el problema y va restando y sumando de uno en uno comprobando que siempre sea mayor que 0. Para la comprobación he definido el predicado derivado *(seatsLeft ?a -aircraft)* que es sencillamente un alias para la comprobación anterior.

   Posteriormente, en las primitivas de *board* y *debark* modifico el *board-limit* según corresponda, la comprobación se hace fuera, no dentro de la primitiva.

2. *Permitir varios pasajeros*

   Este paso ya es más complicado, vamos a separar la explicación en varias etapas.

   a) *Estrategia global*

   La idea es tener un único *task*, **goal-task** que es llamada desde el problema sin parámetros, lo primero que hará será llamar a una primitiva que inicializará todas las variables que nos serán de utilidad durante la resolución del problema pero que no forman parte ni de la definición del problema ni de la restricción, después tenemos 3 etapas diferenciadas, la etapa de *board*, la de *fly* y la de *debark*, y se irán llamando recursivamente la anterior a la siguiente.

   $$board \rightarrow fly \rightarrow debark \rightarrow board$$

   Esta estrategia nos permite separar las distintas funcionalidades del problema en sus bloques contenidos mientras que conserva de forma explícita el sentido de la búsqueda en un árbol que realiza en definitiva *htnp*. La heurística implementada al final es que intente llenar el avión todo lo que pueda, priorizando las personas con el mismo destino, incluso echando a alguna si hiciera falta para que entre otra con el destino mayoritario en el avión y dirigirse directamente al destino cuando bien no queden más plazas en el avión o no haya usuarios en tierra con destinos comunes a alguien del avión.

   b) *Inicialización*

   En particular vamos a inicializar en primer lugar, **(onboard-to ?a -aircraft ?c -city)** que lleva la cuenta de cuántas personas hay en cada avión con cualquiera de los destinos, esto nos permitirá escoger más adelante entre ir a por otro pasajero o ir al destino con más interés; y en segundo **(max-to ?a - aircraft ?c - city)** que define el destino más solicitado para un avión en un momento dado, al principio está inicializada a un valor cualquiera pues se calculará más adelante.

c) *Etapa board*

En la etapa de embarcación distinguimos tres métodos, si quedan sitios libres en el avión, **(Seats-left)**, si no quedan sitios libres pero alguien en la ciudad en la que se encuentra el avión pertenece a un grupo de usuarios dentro del avión con el mismo destino mayor que el grupo de un tercer usuario del avión **(No-seats-left-priority)**; esto nos permitirá sacar usuarios —hacer escalas— para que otros más prometedores suban al vuelo; y finalmente el caso base, que llama a la siguiente etapa *(fly-stage)*.

Los dos primeros métodos se llaman recursivamente a si mismos para que puedan embarcar varios usuarios en la misma ciudad —es posible que suba uno que tenga que bajar en una iteración posterior, esta es una de la infinitas mejoras posibles—.

Muchas de las consecuencias de estas tareas y las siguientes en vez de sacarlas a una acción no durativa las incluyo dentro de *:tasks* como una acción *inline*, la idea era agilizar el desarrollo del prototipo y no sacar nada a no ser que sea usado desde varios lugares.

d) *Etapa fly*

La etapa de vuelo consta de tres métodos distintos, el primero para el vuelo indirecto **(Indirect-flight)**, si cabe alguien más en el avión y en otra ciudad hay alguien con el mismo destino que algún usuario dentro del avión vamos a por él; la segunda, el vuelo directo **(Direct-flight**, que con ayuda de la tarea **(fly-max ?a - aircraft)** va directamente al destino más solicitado actualmente en el avión; y por último, **(Empty-plane)** que mueve el avión hacia el sitio donde haya una persona disponible.

En un principio antes de pasar a la siguiente etapa pasaban por ésta todos los aviones disponibles, esto generaba problemas y lo modifiqué para que aprovechando la forma de trabajar con las variables creadas de HTNP, se complete un ciclo completo para cada avión y pasar después al siguiente avión y su respectivo ciclo completo. Sin embargo quedan en el código restos de este diseño anterior que no he retirado por cuestión de tiempo.

e) *Etapa debark*

La etapa de desembarco consta de cuatro métodos, el primero, **(At-destination)** desembarca a un usuario que se encuentre en su destino y se vuelve a llamar recursivamente, **(Base)** y **(Continue)** que controlan la lógica de pasar a la siguiente etapa si quedan usuarios con destinos pendiente; y **(Finish)** que finaliza la planificación si no quedan destinos por satisfacer.

3. *Limitar tiempo del avión*

Este requisito es más sencillo, inicializo la función **(etime ?a -aircraft)** en la etapa de inicialización por cada avión, se incrementan en cada acción

durativa con el mismo valor que la duración de ésta; y en los predicados derivados *(llega-slow ?a ?c1 ?c2)* y *(llega-fast ?a ?c1 ?c2)* ya no sólo compruebo la capacidad del tanque y el límite de gasolina sino también que **(maxtime ?a)** sea mayor que *(etime ?a)* más la duración del trayecto.

El problema con esta modificación es que provoca un fallo de ejecución a no ser que lo ejecute en modo *verbose* 3, así que lo tengo comentado, para permitir ejecuciones más rápidas excepto en las comprobaciones para este apartado en concreto. Probablemente el error se deba al uso de wine porque la ejecución debería ser consistente en modo verboso y en modo normal. Como en modo verboso la ejecución discurre tal y como se esperaría entiendo que el problema es ajeno a mí.

4. *Mejoras pendientes*

   El programa es tremendamente rudimentario, la programación funcional tiene sus fuertes pero es difícil aplicar todos los conocimientos y desarrollos propios de la programación imperativa a ésta primera. Además, PDDL no está pensado para el uso intensivo de valores numéricos y todo se siente como buscarle trampas al programa para poder optimizar cantidades. Por ejemplo, esto que lo podríamos conseguir con un A* utilizando por ejemplo el número de pasos hasta la situación actual y el número de destinos pendientes, en HTNP la traducción no es cómoda ni mucho menos inmediata, sin embargo, no tener que implementar nosotros la búsqueda dentro del árbol es fantástico, y la salida verbosa es de gran ayuda para la depuración. Al final nos tenemos que conformar con una especie de búsqueda *backtracking* dirigida por un *greedy* y afortunadamente los resultados no son malos.

   Para esta práctica he intentado poner en una balanza una solución aceptable del problema y un uso razonable de mi tiempo, quedan cosas por mejorar que dejo pendientes para el futuro, en primer lugar, priorizar el **orden de embarque**, ahora mismo puede darse el caso en el que entra una persona al avión y antes de despegar entra otra que le echa porque es más interesante para la heurística, esto consume tiempo y no es admisible en un entorno real; en segundo lugar, HTNP y PDDL no se llevan bien con los números porque compaginar lógica de predicados y aritmética es un reto complicado, todo lo que me he propuesto conseguir con números —como el destino con más pasajeros interesados— se siente como un *hack* cutre y que consume muchísimo tiempo desarrollar, me habría gustado optimizar las distancias entre las ciudades, por ejemplo cambiando la selección del destino para el vuelo directo del más demandado como está ahora a una situación intermedia, como el número de kilómetros dividido por el número de interesados y quedarme con el menor, sin embargo, aunque esta modificación es inmediata, el resto de la heurística no estaba pensada para esta estrategia y daba resultados peores porque intentaba llenar mucho el avión de un destino que no le interesaba, así que la he descartado por cuestión de tiempo. Y por último, la gran tarea pendiente

es limpiar el código, quedan restos de estrategias abortadas por todo el código, que sería un trabajo de arqueología eliminar y el programa funciona bien ahora mismo por una mezcla de gracia divina y andamios de bambú, así que no me he atrevido a meterle mano.

Experimentos

Aunque durante el desarrollo parecía ir todo funcionando y cumpliendo las restricciones no soy capaz de poner en pie los experimentos con las condiciones propuestas. Hay al menos dos errores en el código que no encuentro, el primero una condición de salida que permite terminar los planes sin satisfacer todos los objetivos, y otro que agota la pila y produce un error en el intérprete. Adjunto un único experimento con 20 usuarios donde no hay restricciones de gasolina ni de tiempo, sólo de número de pasajeros.

Código

Ejercicio 1

Recoger pasajero en otra ciudad

```
(:method Case3
        :precondition (and
                (at ?p - person ?c1)
                (at ?a - aircraft ?c2 - city)
                (different ?c1 ?c2)
        )
        :tasks (
                (mover-avion ?a ?c2 ?c1)
                (transport-person ?p ?c)
        )
)
```

Ejercicio 2

Combustible y tamaño de tanque suficiente

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

(:derived
  (llega ?a -aircraft ?c1 - city ?c2 -city)
    (> (capacity ?a) (* (distance ?c1 ?c2)(slow-burn ?a)))
)
```

Mover avión o repostar

```
(:task mover−avion
        :parameters (?a − aircraft ?c1 − city ?c2 −city)
        (:method fuel−suficiente
                :precondition (hay−fuel ?a ?c1 ?c2)
                :tasks (
                        (fly ?a ?c1 ?c2)
                )
        )
        (:method fuel−insuficiente
                :precondition (and
                        (llega ?a ?c1 ?c2)
                        (not (hay−fuel ?a ?c1 ?c2))
                )
                :tasks (
                        (refuel ?a ?c1)
                        (mover−avion ?a ?c1 ?c2)
                )
        )
)
```

Ejercicio 3

Nuevo hay-fuel y llega

```
(:derived
  (hay−fuel−fast ?a − aircraft ?c1 − city ?c2 − city)
        (>= (fuel ?a) (∗ (distance ?c1 ?c2)(fast−burn ?a)))
)
(:derived
  (hay−fuel−slow ?a − aircraft ?c1 − city ?c2 − city)
        (>= (fuel ?a) (∗ (distance ?c1 ?c2)(slow−burn ?a)))
)

(:derived
  (llega−fast ?a −aircraft ?c1 − city ?c2 −city)
    (and
      (>= (capacity ?a) (∗ (distance ?c1 ?c2)(fast−burn ?a)))
      (>= (fuel−limit) (+(total−fuel−used) (∗ (distance ?c1 ?c2)(fast−burn
          ↪ ?a))))
    )
)

(:derived
  (llega−slow ?a −aircraft ?c1 − city ?c2 −city)
```

```
    (and
      (>= (capacity ?a) (* (distance ?c1 ?c2)(slow-burn ?a)))
      (>= (fuel-limit) (+(total-fuel-used) (* (distance ?c1 ?c2)(slow-burn
          ↪ ?a))))
    )
)
```

Mover avión

```
(:task mover-avion
  :parameters (?a - aircraft ?c1 - city ?c2 -city)
  (:method fuel-suficiente-fast
    :precondition (and
        (hay-fuel-fast ?a ?c1 ?c2)
        (llega-fast ?a ?c1 ?c2)
    )
    :tasks (
          (zoom ?a ?c1 ?c2)
         )
  )
  (:method fuel-insuficiente-fast
    :precondition (and
      (llega-fast ?a ?c1 ?c2)
      ;(not (hay-fuel-slow ?a ?c1 ?c2))
    )
    :tasks (
        (refuel ?a ?c1)
        (mover-avion ?a ?c1 ?c2)
    )
  )
  (:method fuel-suficiente-slow
    :precondition (and
        (hay-fuel-slow ?a ?c1 ?c2)
        (llega-slow ?a ?c1 ?c2)
    )
    :tasks (
          (fly ?a ?c1 ?c2)
         )
  )
  (:method fuel-insuficiente
    :precondition (and
      (llega-slow ?a ?c1 ?c2)
    )
    :tasks (
        (refuel ?a ?c1)
        (mover-avion ?a ?c1 ?c2)
```

```
    )
  )
)
```

Ejercicio 4

Resultados

Ejercicio 1

```
: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: 12
Generated nodes: 24
Inferences: 0
Time in seconds: 0.02
Real Time: 0.0015
Used Time: 1.78814e−09
System Time: 0.02
```

Ejercicio 2

```
: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 (**refuel** a1 c5) start: 28/06/2007 15:00:00 end: 04/07/2007 21:00:00
:action (**fly** a1 c5 c3) start: 04/07/2007 21:00:00 end: 05/07/2007 12:00:00
:action (**board** p3 a1 c3) start: 05/07/2007 12:00:00 end: 05/07/2007 13:00:00
:action (**refuel** a1 c3) start: 05/07/2007 13:00:00 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: 16 (16)
Expansions: 16
Generated nodes: 32
Inferences: 0
Time **in** seconds: 0.02
Real Time: 0.007625
Used Time: 0.02
System Time: 1.3411e−09

Ejercicio 3

:action (**refuel** a1 c4) start: 05/06/2007 08:00:00 end: 09/06/2007 12:00:00
:action (**zoom** a1 c4 c1) start: 09/06/2007 12:00:00 end: 09/06/2007 20:00:00
:action (**board** p1 a1 c1) start: 09/06/2007 20:00:00 end: 09/06/2007 21:00:00
:action (**refuel** a1 c1) start: 09/06/2007 21:00:00 end: 22/06/2007 09:00:00
:action (**zoom** a1 c1 c5) start: 22/06/2007 09:00:00 end: 22/06/2007 14:00:00
:action (**debark** p1 a1 c5) start: 22/06/2007 14:00:00 end: 22/06/2007
↪ 15:00:00
:action (**refuel** a1 c5) start: 22/06/2007 15:00:00 end: 30/06/2007 23:00:00
:action (**zoom** a1 c5 c2) start: 30/06/2007 23:00:00 end: 01/07/2007 07:00:00
:action (**board** p2 a1 c2) start: 01/07/2007 07:00:00 end: 01/07/2007 08:00:00
:action (**refuel** a1 c2) start: 01/07/2007 08:00:00 end: 13/07/2007 20:00:00
:action (**zoom** a1 c2 c5) start: 13/07/2007 20:00:00 end: 14/07/2007 04:00:00
:action (**debark** p2 a1 c5) start: 14/07/2007 04:00:00 end: 14/07/2007
↪ 05:00:00
:action (**refuel** a1 c5) start: 14/07/2007 05:00:00 end: 26/07/2007 17:00:00
:action (**fly** a1 c5 c3) start: 26/07/2007 17:00:00 end: 27/07/2007 08:00:00
:action (**board** p3 a1 c3) start: 27/07/2007 08:00:00 end: 27/07/2007 09:00:00
:action (**fly** a1 c3 c5) start: 27/07/2007 09:00:00 end: 28/07/2007 00:00:00
:action (**debark** p3 a1 c5) start: 28/07/2007 00:00:00 end: 28/07/2007
↪ 01:00:00
Number of actions: 17 (17)
Expansions: 20
Generated nodes: 39

Inferences: 0
Time **in** seconds: 0.02
Real Time: 0.01
Used Time: 0.02
System Time: 8.9407e−10

Ejercicio 4

:action (add−destination−counter) start: 05/06/2007 08:00:00 end:
↪ 05/06/2007 08:00:00
:action (**board** p16 a1 sevilla) start: 05/06/2007 08:00:00 end: 05/06/2007
↪ 09:00:00
:action (**board** p17 a1 sevilla) start: 05/06/2007 09:00:00 end: 05/06/2007
↪ 10:00:00
:action (**board** p18 a1 sevilla) start: 05/06/2007 10:00:00 end: 05/06/2007
↪ 11:00:00
:action (**board** p19 a1 sevilla) start: 05/06/2007 11:00:00 end: 05/06/2007
↪ 12:00:00
:action (**board** p20 a1 sevilla) start: 05/06/2007 12:00:00 end: 05/06/2007
↪ 13:00:00
:action (**refuel** a1 sevilla) start: 05/06/2007 13:00:00 end: 13/07/2007
↪ 01:00:00
:action (**zoom** a1 sevilla almeria) start: 13/07/2007 01:00:00 end: 13/07/2007
↪ 21:00:00
:action (**debark** p20 a1 almeria) start: 13/07/2007 21:00:00 end: 13/07/2007
↪ 22:00:00
:action (destino−**done** p20 almeria) start: 13/07/2007 22:00:00 end:
↪ 13/07/2007 22:00:00
:action (**debark** p19 a1 almeria) start: 13/07/2007 22:00:00 end: 13/07/2007
↪ 23:00:00
:action (destino−**done** p19 almeria) start: 13/07/2007 23:00:00 end:
↪ 13/07/2007 23:00:00
:action (**board** p1 a1 almeria) start: 13/07/2007 23:00:00 end: 14/07/2007
↪ 00:00:00
:action (**board** p2 a1 almeria) start: 14/07/2007 00:00:00 end: 14/07/2007
↪ 01:00:00
:action (**board** p3 a1 almeria) start: 14/07/2007 01:00:00 end: 14/07/2007
↪ 02:00:00
:action (**board** p4 a1 almeria) start: 14/07/2007 02:00:00 end: 14/07/2007
↪ 03:00:00
:action (**board** p5 a1 almeria) start: 14/07/2007 03:00:00 end: 14/07/2007
↪ 04:00:00
:action (**refuel** a1 almeria) start: 14/07/2007 04:00:00 end: 17/08/2007
↪ 08:00:00

:action (**zoom** a1 almeria granada) start: 17/08/2007 08:00:00 end:
↪ 17/08/2007 16:00:00
:action (**debark** p1 a1 granada) start: 17/08/2007 16:00:00 end: 17/08/2007
↪ 17:00:00
:action (destino−**done** p1 granada) start: 17/08/2007 17:00:00 end:
↪ 17/08/2007 17:00:00
:action (**board** p6 a1 granada) start: 17/08/2007 17:00:00 end: 17/08/2007
↪ 18:00:00
:action (**board** p7 a1 granada) start: 17/08/2007 18:00:00 end: 17/08/2007
↪ 19:00:00
:action (**board** p8 a1 granada) start: 17/08/2007 19:00:00 end: 17/08/2007
↪ 20:00:00
:action (**zoom** a1 granada sevilla) start: 17/08/2007 20:00:00 end: 18/08/2007
↪ 09:00:00
:action (**debark** p16 a1 sevilla) start: 18/08/2007 09:00:00 end: 18/08/2007
↪ 10:00:00
:action (destino−**done** p16 sevilla) start: 18/08/2007 10:00:00 end:
↪ 18/08/2007 10:00:00
:action (**debark** p8 a1 sevilla) start: 18/08/2007 10:00:00 end: 18/08/2007
↪ 11:00:00
:action (destino−**done** p8 sevilla) start: 18/08/2007 11:00:00 end: 18/08/2007
↪ 11:00:00
:action (**debark** p7 a1 sevilla) start: 18/08/2007 11:00:00 end: 18/08/2007
↪ 12:00:00
:action (destino−**done** p7 sevilla) start: 18/08/2007 12:00:00 end: 18/08/2007
↪ 12:00:00
:action (**debark** p6 a1 sevilla) start: 18/08/2007 12:00:00 end: 18/08/2007
↪ 13:00:00
:action (destino−**done** p6 sevilla) start: 18/08/2007 13:00:00 end: 18/08/2007
↪ 13:00:00
:action (**debark** p3 a1 sevilla) start: 18/08/2007 13:00:00 end: 18/08/2007
↪ 14:00:00
:action (destino−**done** p3 sevilla) start: 18/08/2007 14:00:00 end: 18/08/2007
↪ 14:00:00
:action (**debark** p2 a1 sevilla) start: 18/08/2007 14:00:00 end: 18/08/2007
↪ 15:00:00
:action (destino−**done** p2 sevilla) start: 18/08/2007 15:00:00 end: 18/08/2007
↪ 15:00:00
:action (**debark** p18 a1 sevilla) start: 18/08/2007 15:00:00 end: 18/08/2007
↪ 16:00:00
:action (destino−**done** p18 sevilla) start: 18/08/2007 16:00:00 end:
↪ 18/08/2007 16:00:00
:action (**debark** p17 a1 sevilla) start: 18/08/2007 16:00:00 end: 18/08/2007
↪ 17:00:00
:action (destino−**done** p17 sevilla) start: 18/08/2007 17:00:00 end:
↪ 18/08/2007 17:00:00

:action (**refuel** a1 sevilla) start: 18/08/2007 17:00:00 end: 22/09/2007
↪ 05:00:00
:action (**zoom** a1 sevilla madrid) start: 22/09/2007 05:00:00 end: 23/09/2007
↪ 08:00:00
:action (**debark** p5 a1 madrid) start: 23/09/2007 08:00:00 end: 23/09/2007
↪ 09:00:00
:action (destino−**done** p5 madrid) start: 23/09/2007 09:00:00 end:
↪ 23/09/2007 09:00:00
:action (**debark** p4 a1 madrid) start: 23/09/2007 09:00:00 end: 23/09/2007
↪ 10:00:00
:action (destino−**done** p4 madrid) start: 23/09/2007 10:00:00 end:
↪ 23/09/2007 10:00:00
:action (**board** p11 a1 madrid) start: 23/09/2007 10:00:00 end: 23/09/2007
↪ 11:00:00
:action (**board** p12 a1 madrid) start: 23/09/2007 11:00:00 end: 23/09/2007
↪ 12:00:00
:action (**board** p13 a1 madrid) start: 23/09/2007 12:00:00 end: 23/09/2007
↪ 13:00:00
:action (**board** p14 a1 madrid) start: 23/09/2007 13:00:00 end: 23/09/2007
↪ 14:00:00
:action (**board** p15 a1 madrid) start: 23/09/2007 14:00:00 end: 23/09/2007
↪ 15:00:00
:action (**refuel** a1 madrid) start: 23/09/2007 15:00:00 end: 07/11/2007
↪ 02:00:00
:action (**zoom** a1 madrid granada) start: 07/11/2007 02:00:00 end:
↪ 07/11/2007 23:00:00
:action (**debark** p11 a1 granada) start: 07/11/2007 23:00:00 end: 08/11/2007
↪ 00:00:00
:action (destino−**done** p11 granada) start: 08/11/2007 00:00:00 end:
↪ 08/11/2007 00:00:00
:action (**board** p9 a1 granada) start: 08/11/2007 00:00:00 end: 08/11/2007
↪ 01:00:00
:action (**board** p10 a1 granada) start: 08/11/2007 01:00:00 end: 08/11/2007
↪ 02:00:00
:action (**refuel** a1 granada) start: 08/11/2007 02:00:00 end: 13/12/2007
↪ 04:00:00
:action (**zoom** a1 granada sevilla) start: 13/12/2007 04:00:00 end: 13/12/2007
↪ 17:00:00
:action (**debark** p13 a1 sevilla) start: 13/12/2007 17:00:00 end: 13/12/2007
↪ 18:00:00
:action (destino−**done** p13 sevilla) start: 13/12/2007 18:00:00 end:
↪ 13/12/2007 18:00:00
:action (**debark** p12 a1 sevilla) start: 13/12/2007 18:00:00 end: 13/12/2007
↪ 19:00:00
:action (destino−**done** p12 sevilla) start: 13/12/2007 19:00:00 end:
↪ 13/12/2007 19:00:00

:action (**refuel** a1 sevilla) start: 13/12/2007 19:00:00 end: 03/01/2008
↪ 19:00:00
:action (**zoom** a1 sevilla madrid) start: 03/01/2008 19:00:00 end: 04/01/2008
↪ 22:00:00
:action (**debark** p15 a1 madrid) start: 04/01/2008 22:00:00 end: 04/01/2008
↪ 23:00:00
:action (destino−**done** p15 madrid) start: 04/01/2008 23:00:00 end:
↪ 04/01/2008 23:00:00
:action (**debark** p14 a1 madrid) start: 04/01/2008 23:00:00 end: 05/01/2008
↪ 00:00:00
:action (destino−**done** p14 madrid) start: 05/01/2008 00:00:00 end:
↪ 05/01/2008 00:00:00
:action (**refuel** a1 madrid) start: 05/01/2008 00:00:00 end: 18/02/2008
↪ 12:00:00
:action (**zoom** a1 madrid almeria) start: 18/02/2008 12:00:00 end: 19/02/2008
↪ 15:00:00
:action (**debark** p9 a1 almeria) start: 19/02/2008 15:00:00 end: 19/02/2008
↪ 16:00:00
:action (destino−**done** p9 almeria) start: 19/02/2008 16:00:00 end:
↪ 19/02/2008 16:00:00
:action (**refuel** a1 almeria) start: 19/02/2008 16:00:00 end: 05/04/2008
↪ 07:00:00
:action (**zoom** a1 almeria granada) start: 05/04/2008 07:00:00 end:
↪ 05/04/2008 15:00:00
:action (**debark** p10 a1 granada) start: 05/04/2008 15:00:00 end: 05/04/2008
↪ 16:00:00
:action (destino−**done** p10 granada) start: 05/04/2008 16:00:00 end:
↪ 05/04/2008 16:00:00
Number of actions: 78 (156)
Expansions: 118
Generated nodes: 282
Inferences: 0
Time **in** seconds: 0.16
Real Time: 0.165
Used Time: 0.16
System Time: 4.47035e−10