DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

# Técnicas de los Sistemas Inteligentes. Curso 2018-19. Práctica 3: Planificación HTN

# Relación de Ejercicios Prácticos 1: Dominios y problemas de planificación HTN.

## Planificación HTN en el Dominio ZenoTravel

## 1 Realización

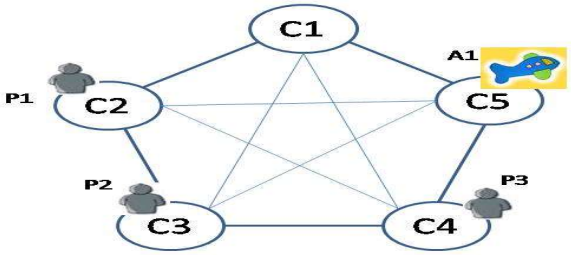
### 1.1 Objetivo

Los ejercicios de esta relación consisten en la definición incremental de un dominio HTN para ser resuelto mediante un algoritmo de planificación HTN en distintas situaciones iniciales. Las acciones primitivas de este dominio están representadas de antemano y el objetivo consiste en definir las acciones compuestas (tareas de alto nivel), métodos de descomposición, predicados, funciones y reglas deductivas necesarias para poder resolver los problemas propuestos. Se propone modelar de forma incremental un dominio HTN para que el planificador pueda resolver problemas en el dominio ZenoTravel.

### 1.2 Dominio ZenoTravel

El dominio ZenoTravel es usado como un estándar para la comparación de técnicas de planificación (Ver http://ipc.icaps-conference.org/ para más información). Es un dominio en el que se representan acciones de transporte aéreo de personas entre distintas ciudades. El dominio permite representar la distancia entre las ciudades, la posición en la que se encuentra cada persona y cada avión. En general, los problemas a resolver en este dominio consisten en encontrar la secuencia adecuada de operaciones de transporte (vuelo, embarque y desembarque) de personas, inicialmente situadas en ciudades, utilizando aviones, también inicialmente situados en ciudades.

Para la primera parte de esta práctica usaremos un ejemplo en el que hay 5 ciudades y un avión. El número de personas dependerá del problema concreto a resolver.




En esta figura se muestra un posible estado inicial en el que las personas P1, P2 y P3 se encuentran en las ciudades C2,C3 y C4, y el avión A1 inicialmente está en C5.

Las acciones primitivas de que se disponen en el dominio ZenoTravel son las siguientes:

1. **Embarcar** una *persona* en un *avión* en una *ciudad* concreta.
2. **Desembarcar** una *persona* en un *avión* en una *ciudad* concreta.
3. **Volar** un *avión* de una *ciudad origen* a una *ciudad destino* a una *velocidad lenta*
4. **Volar** un *avión* de una *ciudad origen* a una *ciudad destino* a una *velocidad rápida*

El dominio también contempla el gasto de fuel que se realiza, además considera que el gasto de fuel no es el mismo para un avión a una velocidad lenta que a una velocidad rápida. Por tanto, como existe un **recurso** (fuel) que se puede agotar, el dominio también contempla la siguiente acción:

- **Repostar** un *avión* en una *ciudad.*

La representación pddl de estas acciones primitivas está en el fichero Primitvas-ZenoTravel.pddl. que puede descargarse desde PRADO del material para esta relación.

### 1.2.1 Embarcar una persona en un avión en una ciudad concreta.

```
(:durative-action board
 :parameters (?p - person ?a - aircraft ?c - city)
 :duration (= ?duration (boarding-time))
 :condition (and  (at ?p ?c)
                  (at ?a ?c))
 :effect (and  (not (at ?p ?c))
               (in ?p ?a)))
```

La representación de esta acción incluye su nombre, parámetros (especificados como variables con tipo) y los siguientes campos:

- `:duration`: donde se representa la duración de la acción (un valor numérico) mediante una expresión de la forma (= `?duration <expresión_aritmética>`). La expresión aritmética es una combinación de operadores aritméticos en la que los operandos son **funciones** de PDDL. Una función de PDDL es un predicado (que puede no tener parámetros como en este ejemplo) que devuelve un valor numérico. Este valor numérico tiene que ser inicializado en el estado inicial (siguiendo la sintaxis `(= <función> <valor>)`, `(= (boarding-time) 1)` por ejemplo en este caso) y que puede ser modificado sólo en los efectos de una acción (ver más abajo un ejemplo en la acción fly sobre cómo modificar el valor de un fluent). La función `(boarding-time)` juega en este caso el papel de una función que siempre devuelve un valor constante.
- `:condition`: representa las precondiciones de la acción como una expresión lógica en la que los operadores son predicados
- `:effect`: representa los efectos de la acción.

La representación de esta acción se interpreta como sigue:

> Si en el estado actual es cierta la expresión lógica descrita en :condition (es decir si existe una unificación entre los predicados y variables de :condition y los predicados y constantes del estado actual de planificación) la acción puede ejecutarse y tiene como efecto la expresión lógica descrita en :effects. El planificador representa internamente, en su proceso de razonamiento, la ejecución de la acción eliminando del estado actual los predicados negados en :effects y añadiendo al estado actual el resto de predicados.

En el caso concreto de esta acción, si existe una unificación para los predicados

- `(at ?p ?c)`, una persona ?p se encuentra en la ciudad ?c
- `(at ?a ?c)`, un avión ?a se encuentra en la misma ciudad ?c

Entonces la ejecución de la acción producirá el siguiente efecto:

- `(not (at ?p ?c))`, la persona ya no está en la ciudad
- `(in ?p ?a)`, la persona está dentro del avión.

Además, el planificador considera también la duración de la acción para llevar a cabo razonamiento temporal sobre restricciones de tiempo (esta característica, el razonamiento temporal, no va a ser considerada para la realización de esta práctica pero se ha preferido mantener la representación de la duración temporal de acciones para que el alumno conozca al menos esta posibilidad).

### 1.2.2 Desembarcar una persona en un avión en una ciudad concreta.

```
(:durative-action debark
 :parameters (?p - person ?a - aircraft ?c - city)
 :duration (= ?duration (debarking-time))
 :condition (and (in ?p ?a)
                 (at ?a ?c))
 :effect (and  (not (in ?p ?a))
               (at ?p ?c)))
```

La explicación anterior es también aplicable a esta acción. (*debarking-time)* es una función con un valor constante que representa el tiempo que tarda una persona en desembarcar de un avión.

### 1.2.3 Volar un avión de una ciudad origen a una ciudad destino a una velocidad lenta

```
(:durative-action fly
 :parameters (?a - aircraft ?c1 ?c2 - city)
 :duration (= ?duration (/ (distance ?c1 ?c2) (slow-speed ?a)))
 :condition (and (at ?a ?c1)
                 (>= (fuel ?a)(* (distance ?c1 ?c2) (slow-burn ?a))))
 :effect (and  (not (at ?a ?c1))
               (at ?a ?c2)
               (increase (total-fuel-used)
                         (* (distance ?c1 ?c2) (slow-burn ?a)))
               (decrease (fuel ?a)
                         (* (distance ?c1 ?c2) (slow-burn ?a)))))
```

La duración de esta acción se calcula a partir de la distancia que separa las dos ciudades y de la función que devuelve el valor de "velocidad lenta" de un avión, definido por la función *(slow-speed ¿a)* e inicializado en el estado inicial.

Esta acción podrá ejecutarse si el avión está en la ciudad origen y la cantidad de fuel actual del avión ?a es suficiente para realizar un vuelo entre las dos ciudades. La cantidad de fuel de un avión ?a está definida también como una función (fuel ?a). El valor de esta función se asigna en el estado inicial y su valor va cambiando a medida que se planifican acciones de vuelo. La cantidad de fuel que gasta un avión se calcula como la razón entre la distancia que separa ambas ciudades (representada con la función *distance*) y la velocidad lenta del avión (representada con la función *slow-speed*).

Los efectos de la acción se interpretan como sigue:

- El avión no está en la ciudad origen y sí está en la destino.

- La cantidad total de fuel usado por el avión se incrementa usando la expresión PDDL (`increase <funcion> <expresion_aritmetica>`) en función de la distancia y velocidad del avión
- La cantidad actual de fuel que contien el avión se decrementa usando la expresión PDDL (`decrease <funcion> <expresion_aritmetica>`) en función de la distancia entre las ciudades y de la relación de consumo de fuel por unidad de distancia (representada por la función *slow-burn*)

En definitiva, los efectos de las acciones pueden usarse también para modificar el valor de las funciones definidas en el dominio. El nuevo valor calculado en los efectos sustituye, en el estado resultante de ejecutar la acción, al antiguo valor que tuviera la función.

### 1.2.4 Volar un avión de una ciudad origen a una ciudad destino a una velocidad rápida

```
(:durative-action zoom
 :parameters (?a - aircraft ?c1 ?c2 - city)
 :duration (= ?duration (/ (distance ?c1 ?c2) (fast-speed ?a)))
 :condition (and (at ?a ?c1)
                 (>= (fuel ?a) (* (distance ?c1 ?c2) (fast-burn ?a))))
 :effect (and (not (at ?a ?c1))
              (at ?a ?c2)
              (increase (total-fuel-used)
                          (* (distance ?c1 ?c2) (fast-burn ?a)))
              (decrease (fuel ?a)
                          (* (distance ?c1 ?c2) (fast-burn ?a)))))
```

La explicación anterior es aplicable a esta acción, con la diferencia de que la "velocidad rápida" implica usar las funciones *fast-burn* y *fast-speed.* En resumen, es una manera de representar que a una mayor velocidad el consumo de fuel es mayor en un grado definido por *farst-burn.*

### 1.2.5 Repostar un avión en una ciudad.

```
(:durative-action refuel
:parameters (?a - aircraft ?c - city)
:duration (= ?duration (/(- (capacity ?a) (fuel ?a))(refuel-rate ?a)))
:condition (and  (>(capacity ?a) (fuel ?a))
                 (at ?a ?c))
:effect (assign (fuel ?a) (capacity ?a)))
```

La acción de repostaje puede realizarse en cualquier ciudad y para que se pueda realizar el avión tiene que estar en una ciudad y la capacidad del avión tiene que ser mayor que el fuel que actualmente tiene. El efecto consiste en volver a reasignar all fuel actual del avión la capacidad que éste tiene. Usando la expresión PDDL `(assign <funcion> <expresion- aritmetica>)`

### 1.2.6 Tareas compuestas y métodos de descomposición

En el fichero zenotravel-V00.pddl se encuentra una descripción de un dominio simple que se entrega al alumno para conocer una primera aproximación al domino ZenoTravel. En la mayor parte de este dominio se sigue la sintaxis estándar PDDL y la descripción de tareas y métodos en una extensión llamada HPDL realizada por el Grupo de Investigación en Sistemas Inteligentes del Departamento de Ciencias de la Computación e I.A. Los aspectos a destacar en este dominio son los siguientes:

- **Hay una parte de preámbulo para configurar las capacidades de razonamiento del planificador que siempre es la misma para todos los dominios**

```
(:requirements
  :typing
  :fluents
 :derived-predicates
 :negative-preconditions
 :universal-preconditions
 :disjuntive-preconditions
 :conditional-effects
 :htn-expansion
  ; Requisitos adicionales para el manejo del tiempo
 :durative-actions
 :metatags
)
```

- **Después viene la declaración de tipos constantes, predicados y funciones que se van a usar en el dominio**

```
(:types aircraft person city - object)
(:constants slow fast - object)
(:predicates (at ?x - (either person aircraft) ?c - city)
             (in ?p - person ?a - aircraft)
             (diferente ?x ?y) ) ;;predicado derivado, ver más abajo
             (igual ?x ?y) ;;predicado derivado, ver más abajo
             (hay-fuel ?a ?c1 ?c2) ;;predicado derivado, ver más abajo
             )
```

```
(:functions (fuel ?a - aircraft) ;;cantidad de fuel actual de un avión
            (distance ?c1 - city ?c2 - city) ;;distancia entre dos ciudades
            (slow-speed ?a - aircraft); ;;velocidad "lenta" de un avión
            (fast-speed ?a - aircraft)  ;;velocidad "rápida" de un avión
            (slow-burn ?a - aircraft);;razón de consumo de un avión a velocidad lenta
            (fast-burn ?a - aircraft);razón de consumo de un avión a velocidad rápida
            (capacity ?a - aircraft) ;;capacidad de fuel de un avión
            (refuel-rate ?a - aircraft) ;;razón de repostaje de un avión (para calcular
                                        ;; el tiempo de repostaje
            (total-fuel-used) ;;valor del fuel total usado
            (boarding-time) ;;valor constante de tiempo de embarque
            (debarking-time) ;;valor constante de tiempo de desembarque
            )
```

- **A continuación se definen reglas de razonamiento para derivar predicados a partir de la información del estado actual:**

```
;; el consecuente "vacío" se representa como "()" y significa "siempre verdad"
;;un objeto es siempre igual a sí mismo
(:derived
  (igual ?x ?x) ())
;; dos objetos son diferentes si no son iguales
(:derived
  (diferente ?x ?y) (not (igual ?x ?y)))

;; El Siguiente literal derivado se utiliza para deducir, a partir de la información en el estado actual,
;; si hay fuel suficiente para que el avión ?a vuele de la ciudad ?c1 a la ?c2
;; el antecedente de este literal derivado comprueba si el fuel actual de ?a es mayor que 1.
;; En este caso es una forma de describir que no hay restricciones de fuel. Pueden introducirse una
;; restricción más compleja  si en lugar de 1 se representa una expresiónn más elaborada (esto es objeto
;; de los siguientes ejercicios).

(:derived
   (hay-fuel ?a - aircraft ?c1 - city ?c2 - city)
   (> (fuel ?a) 1))
```

- **Y finalmente viene la descripción de tareas compuestas y métodos de descomposición**

*;;tarea del nivel de abstracción superior. Esta tarea codifica las diferentes opciones para transportar personas con aviones. Deberá modificarse durante la realización de la práctica.*

```
(:task transport-person
   :parameters (?p - person ?c - city)
   (:method Case1 ; si la persona esá en la ciudad no se hace nada
    :precondition (at ?p ?c)
    :tasks ()
   )
```

*;si la persona no está en la ciudad destino, pero avion y persona están en la misma ciudad*
*;;se embarca a la persona, se mueve el avión (más abajo se muestra como mover avión) y*
*;;desembarca la persona en la ciudad destino*

```
  (:method Case2
    :precondition (and (at ?p - person ?c1 - city)
                       (at ?a - aircraft ?c1 - city))

    :tasks (
             (board ?p ?a ?c1)
             (mover-avion ?a ?c1 ?c)
             (debark ?p ?a ?c )))
   )
```

*;;La siguiente tarea compuesta tiene un único método. Este método se escogerá para usar la acción fly siempre que el avión tenga fuel para volar desde ?c1 a ?c2. Si no hay fuel suficiente el método no se aplicará y la descomposición de esta tarea se intentará hacer con otro método. Cuando se agotan todos los métodos posibles, la descomponsición de la tarea mover-avión "fallará". En consecuencia HTNP hará backtracking y escogerá otra posible vía para descomponer la tarea mover-avion (por ejemplo, escogiendo otra instanciación para la variable ?a).*

```
(:task mover-avion
:parameters (?a - aircraft ?c1 - city ?c2 -city)
(:method fuel-suficiente
  :precondition (hay-fuel ?a ?c1 ?c2)
  :tasks (
         (fly ?a ?c1 ?c2)
        )
   )
  )
```

- **Después de la descripción de las tareas compuestas y métodos viene la descripción de las acciones primitivas**. En este caso, pueden escribirse directamente tal y como vienen en el fichero Primitivas-Zenotravel.pddl o bien se puede poner la expresión

```
(:import "Primitivas-Zenotravel.pddl")
```

para incluir en el dominio el fichero de tareas primitivas (tiene que estar en el mismo directorio que el fichero que contiene la descripción de tareas compuestas) y simplificar el fichero pddl , especificando que se van a utilizar las primitivas contenidas en el fichero.

# 2 Instalación del Planificador

Para poder utilizar HTNP, descargar el fichero htnp.zip y descomprimirlo.

## 2.1 Ejecutar HTNP en Windows

Una vez descomprimido para ejecutar el planficador abrir una ventana de comando de windows (e ir a la directorio donde se haya descomprimido). La sintaxis para invocar al planificador HTNP desde la línea de órdenes en la shell es la siguiente:

Sintaxis: .htnp [opciones] --domain_file (-d) <domain.pddl> --problem_file (-p) <problem.pddl>

Donde las opciones son las siguientes:

- --help (-h): Pantalla de ayuda.
- --debug (-g): Lanza el planificador en modo de depuración del dominio.
- --verbose (-v[<level>] i.e: -v1): Lanza el planificador en modo verbosidad, para imprimir por pantalla las distintas decisiones que va tomando el planificador. Hay tres niveles de verbosidad (1,2 o 3) que aumentan progresivamente el número de mensajes que se imprimen por pantalla. Por defecto es 2.
- --output_file (-o) {filename}: Escribe el plan resultante como un fichero de texto plano en el fichero pasado como argumento.
- --xml_file (-x) {filename}: Escribe el plan resultante en formato xml para su posterior postproceso en el fichero pasado como argumento.
- --expansions_limit <number> : Número máximo de expansiones (aplicaciones de métodos) permitido. Sirve como límite del backtracking permitido durante la búsqueda.
- --depth_limit <number> : Nivel máximo de profundidad en el árbol de expansión permitido durante la búsqueda de un plan.
- --time_limit <number> : Tiempo máximo permitido para la búsqueda de un plan.

### 2.1.1 Ejemplos de dominio y problemas

Para hacer pruebas iniciales con HTNP se suministran dos ficheros (descargarlos desde la página web de las prácticas) con un dominio ejemplo (dominio-bloques.pddl) y un problema (problema-bloques.pddl)

representados en el lenguaje de planificación HTN-PDDL, también puedes consultar el manual sobre HTN-PDDL escrito por Oscar Jesús García Pérez. (descargarlo de la página web).

- Descargar estos ficheros en el mismo directorio de htnp
- Ejecutar: `htnp -d dominio-bloques.pddl -p problema-bloques.pddl` y observar que en la salida producida aparecen las acciones del plan ordenadas.
- Para obtener el plan en un fichero de texto ejecutar `htnp -d dominio-bloques.pddl -p problema-bloques.pddl -o <fichero_de_salida>`
- Para obtner el plan en formato xml (y visualizarlo después en un editor xml) ejecutar `htnp -d dominio-bloques.pddl -p problema-bloques.pddl -x <fichero_de_salida>`

En caso de definir dominios y problemas más complejos, es posible realizar una ejecución del planificador controlada, mediante un depurador integrado. Ver para más detalles el Manual del depurador integrado de HTNP

# 3 Metodología

El trabajo a realizar consistirá en dos partes:

1. Probemas 1 a 3 (4 puntos en total sobre 10). Elaborar un dominio HTN, utilizando el lenguaje de planifiicacón HPDL, de forma incremental con el que puedan resolverse, utilizando el planificador HTNP, los problemas que se describen más abajo.

2. Problema 4 (6 puntos en total sobre 10). Elaborar un dominio siguiendo lo descrito en el problema 4. Este dominio se evaluará sobre un problema con varias personas y varios aviones. La calificación dependerá de criterios de calidad de los planes referidos a: longitud del plan, duración total del plan y fuel consumido en el plan.

### 3.1.1 Comprobar la generación del plan con el problema ejemplo

En el fichero problema-zeno-0.pddl (descargarlo desde PRADO) se describe una situación inicial y un objetivo a alcanzar. Este fichero contiene **un preámbulo de configuración acotado por la palabra clave :customize que debe mantenerse en todos los problemas**. En la situacion inicial se definen las posiciones iniciales de las personas y el avión, además de las distancia entre las ciudades. En este primer problema se asume que tanto la cantidad inicial de fuel como la capacidad de fuel del avión es infinita (se le asigna un valor de 100000, valor que supera considerablemente la cantidad de fuel para resolver el problema).

El problema se define en términos de las tareas compuestas definidas en el dominio (como es necesario en cualquier problema a resolver con técnicas HTN). En este caso concreto el problema consiste en transportar P1 a la ciudad C4, P2 a C5 y p3 a C2.

Para este dominio simple, como no hay restricciones de fuel, en el plan final no se emplean acciones de repostaje.

### 3.1.2 Problema 1

El siguiente problema a resolver (problema-zeno-V01.pddl) 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. La descripción de este problema puede descargarse desde PRADO.

### 3.1.3 Problema 2

El problema 2 (fichero problema-zeno-V02.pddl) 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. La descripción de este problema puede descargarse desde PRADO.

### 3.1.4 Problema 3

Este problema (fichero problema-zeno-V03.pddl) 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. La descripción de este problema puede descargarse de PRADO.

### 3.1.5 Problema 4

Representar un dominio, extendiendo el obtenido en el último problema, con las siguientes características:

1. Modificar la representación de acciones *board* y *debark* para poder representar que cada avión tiene una capacidad máxima de pasajeros, que cada vez que se embarca/desembarca el número de pasajeros de un avión se incrementa/decrementa en 1.
2. Añadir/modificar las tareas compuestas necesarias para que puedan embarcarse varios pasajeros en un avión, una vez que el avión esté en la ciudad adecuada, o desembarcar varios pasajeros de un avión, una vez el avión haya llegado al destino del pasajero.
    a. Pista: es necesario usar un nuevo predicado *(destino ?x – person?y – city)* para informar al planificador del destino de cada persona. Esto significa que por cada *task*

(transport-person ?x ?y) del *goal-task* debe aparecer en el estado inicial un predicado (destino ?x ?y).

b. Pista: para esta mejora es necesario representar tareas compuestas recursivas. Ver el ejemplo de dominio con tareas recursivas en el material de la práctica para obtener una pequeña guía para resolver este problema.

3. Añadir/modificar las tareas primitivas, compuestas, predicados y/o funciones para poder representar que hay una duración limitada para los viajes de cualquier avión.

Experimentar con varios problemas de planificación para este dominio, basados en la siguiente matriz de distancias reales entre aeropuertos españoles:

| | Almería | Barcelona | Bilbao | Cádiz | Córdoba | Gibraltar | Granada | Huelva | Jaén | Madrid | Málaga | Seville |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Almería | --- | | | | | | | | | | | |
| Barcelona | 809 | --- | | | | | | | | | | |
| Bilbao | 958 | 620 | --- | | | | | | | | | |
| Cádiz | 463 | 1284 | 1058 | --- | | | | | | | | |
| Córdoba | 316 | 908 | 796 | 261 | --- | | | | | | | |
| Gibraltar | 339 | 1124 | 1110 | 124 | 294 | --- | | | | | | |
| Granada | 162 | 868 | 829 | 296 | 160 | 255 | --- | | | | | |
| Huelva | 505 | 1140 | 939 | 214 | 241 | 289 | 346 | --- | | | | |
| Jaén | 220 | 804 | 730 | 330 | 108 | 335 | 93 | 347 | --- | | | |
| Madrid | 547 | 621 | 395 | 654 | 396 | 662 | 421 | 591 | 335 | --- | | |
| Málaga | 207 | 997 | 939 | 240 | 165 | 134 | 125 | 301 | 203 | 532 | --- | |
| Seville | 410 | 1046 | 933 | 126 | 143 | 201 | 252 | 95 | 246 | 534 | 209 | --- |

Considerar además los siguientes aspectos para representar esos problemas:

1. Representar problemas con varios aviones y varios límites de fuel, de manera que cada avión tiene una cantidad de fuel limitado a 4 viajes entre las 2 ciudades más distanciadas. Hacer experimentos Intentando obtener planes que gasten el menor fuel posible.
2. Representar problemas en los que haya límites de duración para cada uno de los aviones, Hacer experimentos intentando obtener planes lo más cortos posibles en duración.
3. Representar problemas en los que varíe el número de acciones totales para transportar un número considerable de personas (más de 20, por ejemplo), plantear en el dominio estrategias para obtener planes que tengan el menor número posible de acciones.

La evaluación de este ejercicio consistirá en la ejecución del dominio solución para resolver 3 problemas en los que se usarán varios aviones y varias personas en configuraciones diferentes. Se comprobará cada cada problema:

- Si se resuelve o no
- el número de acciones obtenido

- la cantidad de fuel consumida
- la duración del plan.

## 4 Referencias

En la página web de la práctica hay enlaces a manuales sobre el uso de HTNP y HPDL

## 5 Material a entregar.

Al finalizar la práctica será necesario entregar un fichero zip denominado "Practica3.zip" con el siguiente contenido:

1. Tres directorios (E1, E2, E3) por cada uno de los 3 primeros ejercicios, y en cada directorio un fichero "dominio.pddl" en el que venga descrito el dominio HTN con el que resolver cada uno de los 3 problemas propuestos. Si se resuelven todos los problemas, el trabajo para los tres ejercicios se puntúa con un 4, si resuelve 2 problemas con 4* 2/3 y si resuelve solo uno con 4* 1/3. Es obligatorio tener en cuenta los siguientes aspectos:

    a. Cada dominio debe incluir la definición de predicados, funciones, predicados "derived" y tareas compuestas que se estimen oportunas pero no debe contener descripción de ninguna tarea primitiva

    b. La última línea del dominio debe contener la cláusula (:import "Primitivas-Zenotravel.pddl") para poder incorporar, desde un fichero aparte, las acciones (tareas) primitivas

2. Un directorio adicional (E4) con un fichero "dominio2.pddl" conteniendo el dominio pedido en el Problema4. Se pueden adjuntar tantos ficheros de problema como se desee para justificar la experimentación en este dominio. Este ejercicio se valorará de 0 hasta 6 puntos.

3. Un fichero pdf de un tamaño no superior a 5 páginas justificando las decisiones tomadas sobre la codificación de cada dominio y la experimentación realizada para el último ejercicio.

## Fecha de Entrega

Día 1 de Junio, hasta las 23:00.