Técnicas, Entornos y Aplicaciones de Inteligencia Artificial

Práctica 2. Planificación

Objetivos:

- Conocer el formato PDDL. Conocer la funcionalidad de un planificador.
- Ejemplo ZenoTravel (dominio + problema). Evaluar, conocer y modificar.
- Diseñar un nuevo dominio + problema en PDDL y aplicarlo.

Poliformat:

- Artículos sobre lenguaje PDDL 2.1, 1.2
- Ejemplo de Dominio y Problemas: ZenoTravel
- Boletín Practica. Incluye Enunciado problema a resolver.
- Presentación Práctica
- Planificador (lpg para ejecución en Windows + cygwin1.dll).





Estructura de PDDL: Dominio

Se proporcionan varios ejemplos de dominios

```
(define (domain <name>)
(:requirements <:req 1>... <:req n>) ; requisitos necesarios para entender el dominio
(:types <subtype1>... <subtype n> - <type1> <typen>) ; Tipos a usar. Espacios entre -
(:constants <cons1> ... <cons n>) ; definición de constantes (si se van a usar)
                            ; Info sobre el estado del problema
(:predicates <p1> <p2>... ) ; definición de predicados (info proposicional)
(:functions <f1> <f2>... <fn>) ; definición de funciones (info numérica)
                            ; ------ Acciones ------
(:durative-action 1 ; acción con duración
         :parameters (?par1 – <subtype1> ?par2 – <subtype2> ...)
         :duration <value>
         :condition ([and] <condition<sub>1</sub>>... <condition<sub>n</sub>>) ; "at start", "over all", "at end"
         :effect ([and] <effect<sub>1</sub>>... <effect<sub>n</sub>>) ; "at start", "at end"
 (:durative-action n)
                                ; o :action en el caso de acciones sin duración
```



EJEMPLO

Objetos: personas, aviones y ciudades. Las personas pueden embarcar/desembarcar en/de los aviones, que vuelan entre distintas ciudades.

Existen dos formas de volar, una rápida y una lenta con distintos consumos de combustible.

(define (domain zeno-travel)

```
(:requirements :durative-actions :typing :fluents)
(:types aircraft person city - object)
                                                       ; tres tipos de objetos: avión, persona y ciudad
(:predicates (at ?x - (either person aircraft) ?c - city)
                                                                   ; en qué ciudad está una persona o avión
                                                                   ; Otros predicados.....
(:functions (fuel ?a - aircraft)
                                            ; función numérica: nivel de combustible de un avión
            (distance ?c1 - city ?c2 - city)
                                            ; función numérica: distancia entre 2 ciudades
            (boarding-time)
                                            ; función numérica: tiempo que se tarda en embarcar.
                                            ; Otras funciones......
; A continuación vienen las acciones
(:durative-action board
                          ; acción de embarcar
 :parameters (?p - person ?a – aircraft ?c - city)
                                                   ; hay 3 parámetros: persona, avión y ciudad
 :duration (= ?duration (boarding-time))
                                                    ; la duración viene dada por la función "boarding-time"
 :condition (and (at start (at ?p ?c))
                                                   ; dos condiciones: al principio de la acción ("at start")
                  (over all (at ?a ?c)))
                                                   ; la persona tiene que estar en la ciudad; y durante toda la ejecución
                                                   ; Condicion ("over all"): el avión debe permanecer en la ciudad
 :effect
                                       ; se generan dos efectos
       (and (at start (not (at ?p ?c)))
                                           ; al principio de la acción ("at start") la persona deja de estar en la ciudad;
             (at end (in ?p ?a))))
                                           ; al final de la acción ("at end") la persona pasa a estar dentro del avión.
```





Estructura de PDDL: PROBLEMA

```
(define (problem <name>)
  (:domain <name >) ; nombre del dominio al que pertenece este problema
  (:objects \langle obj_1 \rangle - \langle type_1 \rangle ... \langle obj_n \rangle - \langle type_n \rangle) ; objetos y sus tipos. Espacios entre –
  (:init
                                                 ; estado inicial
       ((cate<sub>1</sub>>) ... (cate<sub>i</sub>>)
                                                                             ; parte proposicional. Sin and.
       (= <function<sub>1</sub>> <value<sub>1</sub>>) ... (= <function<sub>n</sub>> <value<sub>n</sub>>)) ; parte numérica (TODAS)
  (:goal
                                                                              ; objetivos
      (and ((<predicate<sub>1</sub>>) ... (<predicate<sub>i</sub>>) ; objetivos proposicionales, con and
             (<operator<sub>1</sub>> <function<sub>1</sub>> <value<sub>1</sub>>) ...
             (<operator<sub>i</sub>> <function<sub>i</sub>> <value<sub>i</sub>>))) ; objetivos numéricos
  (:metric minimize | maximize <expression>)
                                                                  ; opcional, métrica a min/maximizar
                                                                   ; que representa la calidad del plan
```

```
(define (problem ZTRAVEL-1-2) ;nombre del problema
                            ; nombre del dominio – debe corresponderse con el definido en el dominio
(:domain zeno-travel
                                            ; objetos existentes en el problema
(:objects
                     plane1 – aircraft
                      person1 - person
          ; estado inicial. Todos los predicados que se cumplen y valor de todas las funciones.
(:init
           (at plane1 city0)
                                           ; el avión plane1 en city0 – información proposicional
           (= (slow-speed plane1) 198)
                                           ; la velocidad lenta de plane1 – información numérica
           (= (distance city0 city0) 0)
                                           ; distancias entre pares de ciudades...
           (= (distance city0 city1) 678)
           (= (total-fuel-used) 0)
                                           ; valor inicial del combustible acumulado
           (= (boarding-time) 0.3)
                                           ; duración necesaria para embarcar
           (= (debarking-time) 0.6)
                                           ; duración necesaria para desembarcar
(:goal
        (and
                                           ; objetivos a conseguir
           (at plane1 city1)
                                           ; plane1 tiene que acabar en city1 – objetivo proposicional
           (at person1 city0)
                                           ; person1 tiene que acabar en city0
                                           ; person2 tiene que acabar en city2
           (at person2 city2)
           (< (total-fuel-used) 300)
                                           ; ejemplo de objetivo numérico
          minimize (+ (* 4 (total-time)) (* 0.005 (total-fuel-used))))) ; calidad (métrica) a optimizar
(:metric
```

PLANIFICADOR (ejecución desde terminal Windows (necesario cygwin1.dll)

lpg-td (http://zeus.ing.unibs.it/lpg/)

- Búsqueda local heurística que utiliza grafos de planificación como base de estimaciones.
- ¡No determinista!

```
Ejecución: Ipg-td –o dominio.pddl –f problema.pddl –n 1 (*soluciones*)

Solución: TIEMPO: (ACCION PARAMETROS) [DURACIÓN] [COSTE]

0.0003: (POP-UNITARYPIPE S13 B1 A1 A3 B5 LCO OCA1 TA1-1-OCA1 TA3-1-LCO)

[D: 2.0000; C: 0.1000]
```

NOTA: Alternativamente a la opción "—n", también se puede utilizar la opción —speed o —quality. —**speed**: trata de encontrar una solución tan rápidamente como pueda.

-quality: trata de encontrar una solución de buena calidad (aunque sin garantía de optimalidad).

.....; acciones no necesariamente ordenadas en tiempo





Plan Obtenido:

TIEMPO: (ACCION PARAMETROS) [DURACIÓN] [COSTE]

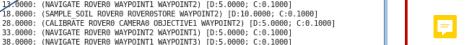
Duracion: Duración de la acción.

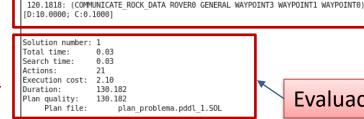
Coste: Coste que supone la ejecución de la acción en la métrica definida para el problema.

```
Solution number: 1
Total time:
Search time:
Actions:
Execution cost:
                 2.10
Duration:
                 130.182
Plan quality:
                130.182
     Plan file:
```

Ejecución lpd-td (rover)

```
Parsing domain file: domain 'ROVER' defined ...
Parsing problem file: problem 'ROVERPROB1234'
                                        Plan quality
Modality: Incremental Planner
                                        (calidad del plan
Number of actions
Number of conditional actions :
Number of facts
                                        según la métrica
Analyzing Planning Problem:
                                        definida en el
      Temporal Planning Problem: YES
      Numeric Planning Problem: YES
      Problem with Timed Initial Litearals:
      Problem with Derived Predicates: NO
                                        problema
Evaluation function weights:
    Action duration 1.00; Action cost 0.00
                                        (por defecto,
Computing mutex... done
                                        duración del plan)
Preprocessing total time: 0.00 seconds
Searching ('.' = every 50 search steps):
```





AYPOINT0) [D:15.0000; C:0.1000]

D:10.0000; C:0.10001

Time: (ACTION) [action Duration; action Cost]

8.0000: (DROP ROVERO ROVEROSTORE) [D:1.0000; C:0.1000]

0.0000: (SAMPLE ROCK ROVERO ROVEROSTORE WAYPOINT3) [D:8.0000; C:0.1000] 8.0000: MAVIGATE ROVERO WAYPOINT3 WAYPOINT1) [D:5.0000; C:0.1000]

43.0000: (NAVIGATE ROVERO WAYPOINT3 WAYPOINT0) [D:5.0000; C:0.1000] 48.0000: (RECHARGE ROVERO WAYPOINTO) [D:7.2727; C:0.1000]

55.2727: (NAVIGATE ROVERO WAYPOINTO WAYPOINT3) [D:5.0000; C:0.1000] 60.2727: (NAVIGATE ROVERO WAYPOINT3 WAYPOINT1) [D:5.0000; C:0.1000]

65.2727: (NAVIGATE ROVERO WAYPOINT1 WAYPOINT3) [D:5.0000; C:0.1000] 70.2727: (NAVIGATE ROVERO WAYPOINT3 WAYPOINT0) [D:5.0000; C:0.1000] 75.2727: (RECHARGE ROVERO WAYPOINTO) [D:2.9091; C:0.1000]

85.1818: (NAVIGATE ROVERO WAYPOINTO WAYPOINT3) [D:5.0000; C:0.1000]

105.1818: (NAVIGATE ROVERO WAYPOINT3 WAYPOINT1) [D:5.0000; C:0.1000] 110.1818: (COMMUNICATE SOIL DATA ROVERO GENERAL WAYPOINT2 WAYPOINT1 WAYPOINTO)

78.1818: (TAKE IMAGE ROVERO WAYPOINTO OBJECTIVE1 CAMERAO HIGH RES) [D:7.0000; C

90.1818: (COMMUNICATE IMAGE DATA ROVERO GENERAL OBJECTIVE1 HIGH RES WAYPOINT3 W

Evaluación



plan problema.pddl 1.SOL

Plan computed:

Práctica a realizar:

Planificación de transporte multimodal de mercancías

Evaluación:

 Realizad el ejercicio propuesto (se necesitará para el día de la evaluación, en el que se plantearán ampliaciones y/o modificaciones)

Calendario:

Sem	<u>LABORATORIO</u>	Evaluación
5-X	Planificación	
19-X	Planificación	
26-X		P2: Planificación

Aplicación y evaluación de Planificación (15%) P2



Notas (FAQ's)

- Inicializar todas las funciones (fluents) y estados de los objetos.
- Expresar bien las **condiciones** al inicio (START), al final (END) y OVERALL de las acciones durativas
- Expresar bien los **efectos** al inicio (START), al final (END) de las acciones durativas
- El planificador LPG no soporta ciertas precondiciones numéricas como "over all" (error: "PRECONDITION TYPE NOT HANDLED YET"). La alternativa es tratar esa precondición mediante predicados.
- En planificador LPG buscad solo unas pocas soluciones (por temas de ejecución). Preferiblemente, solo 1 (-n 1). Además, si se indican más podría NO PARAR!

 Se puede usar también la opción –speed o –quality
- LPG-td no es determinista
- Asegurarse de que el planificador, dominio y problema están en misma carpeta

Probad LPG y analizad los planes solución (y métricas) obtenidos

