# Técnicas, Entornos y Aplicaciones de Inteligencia Artificial

## Práctica 4. Planificación (linux)

**Objetivos:**

- **Conocer el formato PDDL.**
- **Aplicar una colección de dominios y problemas de planificación, en algunos de los planificadores más comunes.**
- **Realizar modificaciones en los archivos PDDL previos.**
- **Diseñar un nuevo dominio+problema en PDDL y aplicarlo.**

## 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 que se usan

**(:constants** <cons1> … <cons n>) ; definición de constantes (si se van a usar)

*; Info sobre estado actual del problema*

**(:predicates** <p1> <p2>… <p n>) ; definición de predicados (info proposicional)

**(:functions** <f1> <f2>… <fn>) ; definición de funciones (info numérica)

*; Acciones u Operadores*

**(:durative-action** 1 *; acción con duración, también denominados operadores…*

:parameters (?par1 – <subtype1> ?par2 – <subtype2> ...)

:duration <value>

:condition (<condition$_1$>… <condition$_n$>) *; "at start", "over all", "at end",*

:effect (<effect$_1$>… <effect$_n$>) *; "at start", "at end"*

)

……

(**:durative-action** n) ; n>0, o **:action** en el caso de acciones sin duración

)

***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) ***; existen 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 para representar el nivel de combustible de un avión***
(distance ?c1 - city ?c2 - city) ***; función numérica para conocer la distancia entre 2 ciudades***
(boarding-time) ***; función numérica para el tiempo que se tarda en embarcar.***
--- ***; Otras funciones……***

**; A continuación vienen las acciones (también denominadas operadores)**

**(: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***
***; ("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** <obj$_1$> - <type$_1$> … <obj$_n$> - <type$_n$>) *; objetos y sus tipos*

**(:init** *; estado inicial*

(<predicate$_1$>) … (<predicate$_i$>) *; parte proposicional*

(= <function$_1$> <value$_1$>) … (= <function$_n$> <function$_n$>)) *; parte numérica (TODAS)*

**(:goal** *; objetivos*

(and ((<predicate$_1$>) … (<predicate$_i$>) *; objetivos proposicionales*

(<operator$_1$> <function$_1$> <value$_1$>) …

(<operator$_j$> <function$_j$> <value$_j$>))) *; objetivos numéricos*

**(:metric** minimize|maximize <expression>) *; opcional, métrica a min/maximizar*

*; que representa la **calidad** del plan*

)

***Se proporcionan varios ejemplos de problemas sobre cada dominio***

```
(define (problem ZTRAVEL-1-2)     ;nombre del problema
(:domain  zeno-travel             ; nombre del dominio – debe corresponderse con el definido en el dominio
(:objects                         plane1 – aircraft        ; objetos existentes en el problema
                                  person1 - person
                                                .....)
(:init          ; estado inicial .  Todos los predicados que se cumplen y  valor de todas  funciones.
                (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
                (at person2 city2)                 ; person2 tiene que acabar en city2
                (< (total-fuel-used) 300)          ; ejemplo de objetivo numérico
                ))

(:metric    minimize (+ (* 4 (total-time))  (* 0.005 (total-fuel-used))))    ; calidad (métrica) a optimizar
```

## EJEMPLOS PDDL (Dominio + Problema)

**Rovers (misiones del Mars Exploration Rover de la NASA).** *40 instancias*

- El objetivo es utilizar rovers para visitar puntos de interés de un planeta y realizar muestreos para, posteriormente, comunicar los datos a su lanzadera.
- Se incluyen restricciones de navegación hacia los puntos de interés, de visibilidad de la lanzadera, de consumo de energía y de capacidad para realizar distintos tipos de muestras.

**Storage.** *30 instancias*

- El objetivo es trasladar cajas desde unos contenedores a depósitos/almacenes utilizando grúas.
- En cada depósito, cada grúa puede moverse siguiendo un determinado mapa espacial que conecta las áreas del depósito.
- Se incluyen restricciones espaciales en los depósitos y zonas de carga, distinto número de depósitos, grúas disponibles, contenedores y cajas.

**Pipes.** *50 instancias*

- Dominio utilizado para controlar el flujo de los derivados del petróleo a través de una red de tuberías.
- Se incluyen restricciones sobre las tuberías, sus segmentos y ocupación, compatibilidad e interferencia entre productos y capacidades de los tanques.

**PLANIFICADORES** ***(ejecución desde terminal linux. Puede requerir "chmod +x PROGRAMA ")***

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

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

**Ejecución:** **./lpg-td-1.0 –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]
……..: ……

## mips-xxl (http://sjabbar.com/mips-xxl-planner)

- Planificador heurístico (determinista) que utiliza grafos de planificación relajados.

**Ejecución:** **./mips-xxl –o dominio.pddl –f problema.pddl –O** (*Optimiza*)

**Solución:** 0.00: (POP-UNITARYPIPE S13 B1 A1 A3 B5 LCO OCA1 TA1-1-OCA1 TA3-1-LCO )
[2.00]
……: ……

## Ejecución lpd-td-1.0 (rover)

```
Parsing domain file:  domain 'ROVER' defined ... done.
Parsing problem file:  problem 'ROVERPROB1234' defined ... done.



Modality: Incremental Planner

Number of actions            :      64
Number of conditional actions :       0
Number of facts              :      36


Analyzing Planning Problem:
        Temporal Planning Problem: YES
        Numeric Planning Problem: YES
        Problem with Timed Initial Litearals: NO
        Problem with Derived Predicates: NO

Evaluation function weights:
     Action duration 1.00; Action cost 0.00


Computing mutex... done

Preprocessing total time: 0.00 seconds

Searching ('.' = every 50 search steps):
.. solution found:

Plan computed:
   Time: (ACTION) [action Duration; action Cost]
 0.0000: (SAMPLE_ROCK ROVER0 ROVER0STORE WAYPOINT3) [D:8.0000; C:0.1000]
 8.0000: (NAVIGATE ROVER0 WAYPOINT3 WAYPOINT1) [D:5.0000; C:0.1000]
 8.0000: (DROP ROVER0 ROVER0STORE) [D:1.0000; C:0.1000]
 13.0000: (NAVIGATE ROVER0 WAYPOINT1 WAYPOINT2) [D:5.0000; C:0.1000]
 18.0000: (SAMPLE_SOIL ROVER0 ROVER0STORE WAYPOINT2) [D:10.0000; C:0.1000]
 28.0000: (CALIBRATE ROVER0 CAMERA0 OBJECTIVE1 WAYPOINT2) [D:5.0000; C:0.1000]
 33.0000: (NAVIGATE ROVER0 WAYPOINT2 WAYPOINT1) [D:5.0000; C:0.1000]
 38.0000: (NAVIGATE ROVER0 WAYPOINT1 WAYPOINT3) [D:5.0000; C:0.1000]
 43.0000: (NAVIGATE ROVER0 WAYPOINT3 WAYPOINT0) [D:5.0000; C:0.1000]
 48.0000: (RECHARGE ROVER0 WAYPOINT0) [D:7.2727; C:0.1000]
 55.2727: (NAVIGATE ROVER0 WAYPOINT0 WAYPOINT3) [D:5.0000; C:0.1000]
 60.2727: (NAVIGATE ROVER0 WAYPOINT3 WAYPOINT1) [D:5.0000; C:0.1000]
 65.2727: (NAVIGATE ROVER0 WAYPOINT1 WAYPOINT3) [D:5.0000; C:0.1000]
 70.2727: (NAVIGATE ROVER0 WAYPOINT3 WAYPOINT0) [D:5.0000; C:0.1000]
 75.2727: (RECHARGE ROVER0 WAYPOINT0) [D:2.9091; C:0.1000]
 78.1818: (TAKE_IMAGE ROVER0 WAYPOINT0 OBJECTIVE1 CAMERA0 HIGH_RES) [D:7.0000; C
:0.1000]
 85.1818: (NAVIGATE ROVER0 WAYPOINT0 WAYPOINT3) [D:5.0000; C:0.1000]
 90.1818: (COMMUNICATE_IMAGE_DATA ROVER0 GENERAL OBJECTIVE1 HIGH_RES WAYPOINT3 W
AYPOINT0) [D:15.0000; C:0.1000]
 105.1818: (NAVIGATE ROVER0 WAYPOINT3 WAYPOINT1) [D:5.0000; C:0.1000]
 110.1818: (COMMUNICATE_SOIL_DATA ROVER0 GENERAL WAYPOINT2 WAYPOINT1 WAYPOINT0)
[D:10.0000; C:0.1000]
 120.1818: (COMMUNICATE_ROCK_DATA ROVER0 GENERAL WAYPOINT3 WAYPOINT1 WAYPOINT0)
[D:10.0000; C:0.1000]

Solution number: 1
Total time:      0.03
Search time:     0.03
Actions:         21
Execution cost:  2.10
Duration:        130.182
Plan quality:    130.182
     Plan file:       plan_problema.pddl_1.SOL
```

*Métrica por defecto:*
*Duración del plan*

## Ejecución mips-xxl (rover)

```
bash-4.1$ ./mips-xxl -o rovers.pddl -f problema.pddl -O

ff: parsing domain file
domain 'ROVER' defined
 ... done.
ff: parsing problem file
problem 'ROVERPROB1234' defined
 ... done.

GTT initialized to 1.000000

metric established (normalized to minimize):

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is  optimization

advancing to distance:    9
                          8
                          7
                          6
                          5
                          4
                          3
                          2
                          1
                          0      New upper bound (minimization assumed)
        Add (< (metric) -0.01) to goal description
        iterate to improve sol. quality

Plan-quality: 10.00

ff: found legal plan as follows

0.00: (SAMPLE-ROCK ROVER0 ROVER0STORE WAYPOINT3 ) [8.00]
8.01: (NAVIGATE ROVER0 WAYPOINT3 WAYPOINT1 ) [5.00]
13.02: (NAVIGATE ROVER0 WAYPOINT1 WAYPOINT2 ) [5.00]
18.03: (CALIBRATE ROVER0 CAMERA0 OBJECTIVE1 WAYPOINT2 ) [5.00]
23.04: (TAKE-IMAGE ROVER0 WAYPOINT2 OBJECTIVE1 CAMERA0 HIGH_RES ) [7.00]
30.05: (COMMUNICATE-IMAGE-DATA ROVER0 GENERAL OBJECTIVE1 HIGH_RES WAYPOINT2 WAYPOINT0 ) [15.00]
45.06: (DROP ROVER0 ROVER0STORE ) [1.00]
46.07: (SAMPLE-SOIL ROVER0 ROVER0STORE WAYPOINT2 ) [10.00]
56.08: (COMMUNICATE-SOIL-DATA ROVER0 GENERAL WAYPOINT2 WAYPOINT2 WAYPOINT0 ) [10.00]
66.09: (COMMUNICATE-ROCK-DATA ROVER0 GENERAL WAYPOINT3 WAYPOINT2 WAYPOINT0 ) [10.00]

  57.05
bash-4.1$
```

La calidad del plan representa la métrica que se ha definido en el problema

*Métrica por defecto:*
*Longitud del plan (nº acciones)*

*Plan Secuencializado*

*Duracion del Plan Paralelizable (ffPSolution.solu))*

# Evaluación:

- Parte 1 (15%). Comparación de planificadores en los dominios proporcionados (rovers, storage, pipes)
  - Ejecutad los dos planificadores sobre los distintos problemas.
  - Usad **solo** los 4 primeros problemas (p01, p02… p04) de cada dominio
  - Evaluad calidad del plan *(nº de acciones, duración "total-time") del plan* y el *tiempo de ejecución* mediante:

    time ./mips-xxl –o dominio.pddl –f problema.pddl –O
  - Resumid los resultados y hacer una comparativa crítica.

- Parte 2 (85%). Definir dominio+problema en PDDL según propuesta del boletín o uno nuevo en particular
  - Probadlo con los 2 planificadores y evaluad los resultados

## Calendario:

**Entregable + Memoria con los resultados obtenidos (15%)**

| Sem | LABORATORIO | Entrega |
|---|---|---|
| 21-XI | Planificación | |
| 28-XI | Planificación | |
| 5-XII | | **4.P: Planificación** |

## Evaluación de los Planificadores:

| | Dominio ROVER | | | | Dominio STORAGE | | | | Dominio PIPES | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| LPG | | | | | | | | | | | | |
| MIPS | | | | | | | | | | | | |

En cada celda:

- Nº acciones
- Tiempo cómputo
- Duración Plan

+ Comentario y Crítica

*En los problemas propuestos:*
*Evaluad la métrica apropiada.*

## Notas (FAQ's)

- Inicializar todas las funciones (fluents)
- Expresar bien las condiciones  al inicio, al final y OVERALL de acciones durativas
- El planificador LPG no soporta ciertas precondiciones numéricas (error: "PRECONDITION TYPE NOT HANDLED YET"). La alternativa es tratar esa precondición mediante predicados.
- En planificador LPG-td, buscar solo 1 solucion (temas de coste). Además, si se indican más y no hay solución, NO PARA!.
- LPG-td no es determinista
- Asegurarse que el planificador, dominio y problema están en misma carpeta
- Para ejecutar en LINUX: chmod + x Programa

**Sobre la Memoria:**

- Modelar correctamente dominio y problema.
- Evaluar casos de prueba.
- Incluir conclusiones sobre los resultados obtenidos