PRÁCTICA 4: PLANIFICACIÓN

Uso de lpg-td y mips-xxl

Pascual Andrés Carrasco Gómez

Índice de contenido

APARTADO 1

1. Consideraciones en la creación de las tablas

- Los valores obtenidos en la tabla para lpg-td se han obtenido realizando 5 ejecuciones para cada problema, ordenándolas según la calidad del plan obtenido y escogiendo la 3 ejecución en ese orden, es decir, la mediana.
- Los valores situados a continuación de lpg-td en las tablas hacen referencia a los valores -n con los que se ha realizado las ejecuciones. Ejemplo: tabla Storage; lpg-td (1,3) hace referencia a las ejecuciones:
  - time ./lpg-td-1.0 -o domain.pddl -f p03.pddl -n 1
  - time ./lpg-td-1.0 -o domain.pddl -f p04.pddl -n 3
- Los valores para mips-xxl se han obtenido ejecutando la opción -O.
- El criterio escogido para comparar soluciones entre lpg-td y mips es la calidad del plan.

2. Tablas con conclusiones individuales

Rovers						
	P01			P02		
	T. Ejecución	N.º Acciones	Total-Time	T. Ejecución	N.º Acciones	Total-Time
lpg-td (2,2)	0.024s	10	75.00	0.020s	8	65.00
mips-xxl	0.008s	10	57.05	0.008s	8	47.04
	P03			P04		
	T. Ejecución	N.º Acciones	Total-Time	T. Ejecución	N.º Acciones	Total-Time
lpg-td (3,3)	0.024s	13	62.00	0.016s	8	50.0
mips-xxl	0.032s	13	67.05	0.012s	8	38.03

P01,P02,P04: Observamos que la solución de mips es mejor ya que la duración del plan (Total-Time) es menor, el numero de acciones son el mismo en ambos casos y el tiempo de ejecución también es mejor el de mips.

P03: Observamos que la solución de lpg-td es mejor ya que la duración del plan es menor, el numero de acciones son el mismo en ambos casos y el tiempo de ejecución también es mejor el de lpg-td.

Storage						
	P01			P02		
	T. Ejecución	N.º Acciones	Total-Time	T. Ejecución	N.º Acciones	Total-Time
lpg-td (1,1)	0.004s	3	3.00	0.008s	3	3.00
mips-xxl	0.000s	3	3.00	0.000s	3	3.00
	P03			P04		
	T. Ejecución	N.º Acciones	Total-Time	T. Ejecución	N.º Acciones	Total-Time
lpg-td (1,3)	0.008s	3	3.00	0.020s	8	10.0
mips-xxl	0.000s	3	3.00	0.004s	8	10.0

P01,P02,P03,P04: Observamos que la duración del plan y el número de acciones es el mismo en ambos casos, el tiempo de ejecución es mejor el de mips.

Pipes						
	P01			P02		
	T. Ejecución	N.º Acciones	Total-Time	T. Ejecución	N.º Acciones	Total-Time
lpg-td (2,3)	0.012s	5	6.00	2.568s	12	22.00
mips-xxl	0.008s	5	6.02	0.024s	14	24.11
	P03			P04		
	T. Ejecución	N.º Acciones	Total-Time	T. Ejecución	N.º Acciones	Total-Time
lpg-td (3,3)	1.144s	9	14.00	2.720s	13	16.00
mips-xxl	0.048s	9	14.06	0.108s	13	22.10

P01,P03: Observamos que la duración del plan y el número de acciones en ambos casos es el mismo, el tiempo de ejecución es mejor el de mips.
P02: Observamos que la solución de lpg-td es mejor ya que la duración del plan y el número de acciones son mejores sin embargo el tiempo de ejecución es mejor el de mips.
P04: Observamos que la duración del plan es mejor en lpg-td, el número de acciones es el mismo en ambos casos y el tiempo de ejecución es mejor el de mips.

3. Conclusiones generales

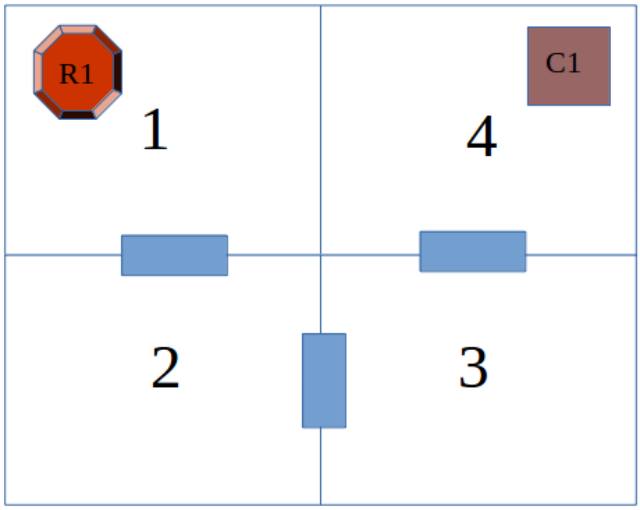
En tiempo de ejecución es mejor el planificador mips-xxl ya que obtiene una solución de forma mas rápida, para el problema Rovers observamos que tomando un criterio basándonos en solo los cuatro primeros problemas el mejor planificador es mips-xxl, en el problema Storage observamos que los resultados obtenidos son los mismos con ambos planificadores (excepto el tiempo de ejecución) y en el problema Pipes observamos que es mejor utilizar lpg-td.

APARTADO 2

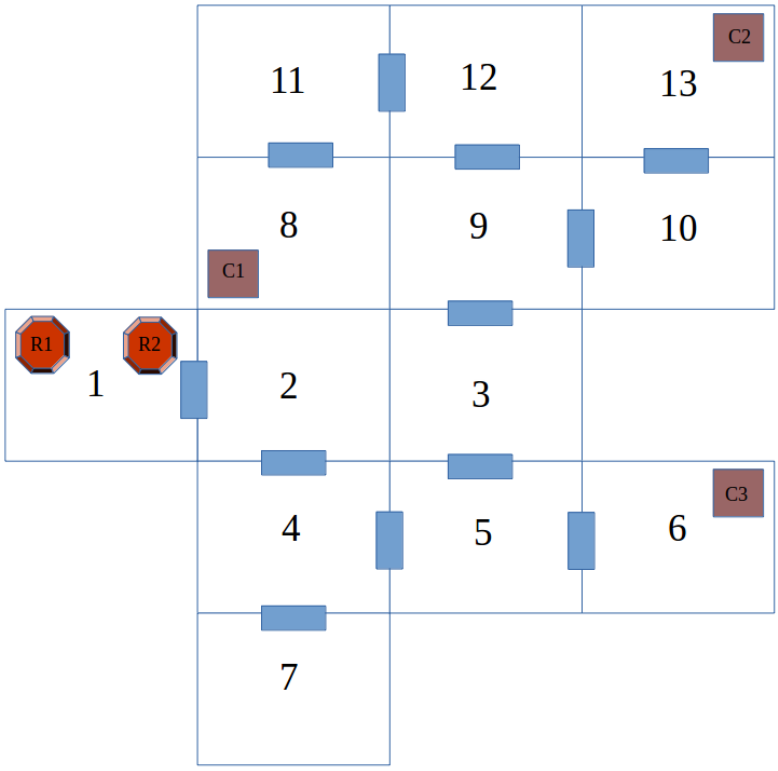
1. Descripción del problema

El problema se compone de un escenario compuesto por habitaciones que están conectadas unas con otras mediante puertas, un conjunto de robots y un conjunto de cajas, los robots parten de la habitación 1 y las cajas están distribuidas por las habitaciones, el problema se define en obtener el conjunto de acciones (un plan) que permitan dejar cada caja en una habitación concreta y los robots en la habitación origen (habitación 1). Vamos a trabajar en dos escenarios:

Uno trivial compuesto por cuatro habitaciones, un robot y una caja:



El segundo escenario es mas complejo ya que esta compuesto por trece habitaciones, dos robots y tres cajas:



2. Estudio del dominio del problema

En este apartado de la memoria vamos a especificar el dominio de nuestro problema, para ello vamos a explicar el archivo “dominio.pddl” paso a paso:

Nombre del dominio y requerimientos:

Como podemos observar nuestro dominio se llama robots-cajas, y los requerimientos que necesita este dominio son acciones durativas, predicados con tipo y funciones numéricas, los cuales vamos a explicar mas adelante.

```
10  ; Nombre del dominio
11  (define (domain robots-cajas)
12
13  ; Se requieren:
14  ;   Acciones durativas
15  ;   Predicados con tipo
16  ;   Funciones numericas
17  (:requirements :durative-actions :typing :fluents)
```

Tipos de objetos:

Nuestro dominio tiene tres tipos de objetos que son el tipo robot, caja y habitación.

```
19  ; Tipos de objetos
20  (:types robot caja habitacion - object)
```

Predicados:

Nuestro dominio tiene cuatro predicados que nos permiten representar información proposicional del estado actual en el problema de planificación, los predicados representan lo siguiente:

- (esta ?x - (either robot caja) ?h - habitacion): Representa por un lado que el robot ?x esta en la habitación ?h y al mismo tiempo nos permite representar que la caja ?x esta en la habitación ?h. NOTA: esta representación es la misma que la siguiente representación,
  (esta ?x - robot ?h - habitacion)
  (esta ?x - caja ?h - habitacion)
- (ocupado ?r - robot ?c - caja): Representa que el robot ?r tiene la caja ?c cargada.
- (libre ?r - robot): Representa que el robot ?r no tiene ninguna caja cargada.
- (puerta ?h1 - habitacion ?h2 - habitacion): Representa que hay una puerta entre la habitación ?h1 y la habitación ?h2.

```
; Predicados
(:predicates (esta ?x - (either robot caja) ?h - habitacion)
             (ocupado ?r - robot ?c - caja)
             (libre ?r - robot)
             (puerta ?h1 - habitacion ?h2 - habitacion))
```

Funciones numéricas:

Nos permiten representar información numérica.

Tenemos seis funciones numéricas que describimos a continuación:

- (velocidad ?r - robot): Es la velocidad a la que funciona el robot ?r.
- (hab-visitadas): Es el número de habitaciones que se visitan en la ejecución del plan.
- (tiempo-cargar-caja): Es el tiempo que tarda un robot en cargar una caja.
- (tiempo-descargar-caja): Es el tiempo que tarda un robot en descargar una caja.
- (mover-robot-caja): Es el tiempo que tarda un robot con una caja cargada en ir de una habitación a otra habitación separadas por una puerta.
- (mover-robot-sin-caja): Es el tiempo que tarda un robot sin una caja cargada en ir de una habitación a otra habitación separadas por una puerta.

```
; Funciones numericas
(:functions
            (velocidad ?r - robot)
            (hab-visitadas)
            (tiempo-cargar-caja)
            (tiempo-descargar-caja)
            (mover-robot-caja)
            (mover-robot-sin-caja))
```

Acciones del problema (Operadores):

Las acciones nos permiten modelar el conocimiento, nuestro dominio tiene cuatro acciones que nos permiten hacer funcionar a los robots dentro de los escenarios moviendo cajas y obteniendo así el objetivo buscado. A continuación se describe cada acción de forma detallada, comentando que parámetros necesita la acción, y sus precondiciones y efectos.

coger-caja:

Los parámetros que necesita la acción son un robot ?r una caja ?c y una habitación ?h.

La acción tiene una duración definida por la función numérica tiempo-cargar-caja.

Como condición al principio de la acción se tiene que cumplir que la caja ?c este en la habitación ?h y que el robot ?r este libre (sin caja). Durante toda la acción el robot ?r tiene que estar en la habitación ?h.

Como efecto al principio de la acción se elimina que la caja ?c esta en la habitación ?h y también se elimina que el robot ?r esta libre (sin caja). Al final de la acción se añade que el robot ?r tiene la caja ?c, es decir, pasa a estar ocupado (con caja).

```
(:durative-action coger-caja
 :parameters (?r - robot ?c - caja ?h - habitacion)
 :duration (= ?duration (tiempo-cargar-caja))
 :condition (and (at start (esta ?c ?h))
                 (at start (libre ?r))
                 (over all (esta ?r ?h)))
 :effect (and (at start (not (esta ?c ?h)))
              (at start (not (libre ?r)))
              (at end (ocupado ?r ?c))))
```

dejar-caja:

Los parámetros que necesita la acción son un robot ?r una caja ?c y una habitación ?h.

La acción tiene una duración definida por la función numérica tiempo-descargar-caja.

Como condición al principio de la acción se tiene que cumplir que el robot ?r este ocupado (con la caja ?c). Durante toda la acción el robot ?r tiene que estar en la habitación ?h.

Como efecto al principio de la acción se elimina que el robot ?r este ocupado (con la caja ?c). Al final de la acción se añade que el robot ?r esta libre (sin caja) y que la caja ?c esta en la habitación ?h.

```
(:durative-action dejar-caja
 :parameters (?r - robot ?c - caja ?h - habitacion)
 :duration (= ?duration (tiempo-descargar-caja))
 :condition (and (at start (ocupado ?r ?c))
                 (over all (esta ?r ?h)))
 :effect (and (at start (not (ocupado ?r ?c)))
              (at end (libre ?r))
              (at end (esta ?c ?h))))
```

mover-robot-vacio:

Los parámetros que necesita la acción son un robot ?r, una habitación origen ?ho y una habitación destino ?hd.

La acción tiene una duración definida por el tiempo que tarda en moverse un robot sin caja de una habitación a otra a través de una puerta partido entre la velocidad a la que trabaja dicho robot.

Como condición al principio se tiene que cumplir que exista una puerta entre la habitación origen ?ho y la habitación destino ?hd y se tiene que cumplir que el robot ?r este en la habitación origen ?ho. Durante toda la acción se tiene que cumplir que el robot ?r este libre (sin caja).

Como efecto al principio de la acción se elimina que el robot ?r este en la habitación origen ?ho. Al final de la acción se añade que el robot ?r esta en la habitación destino ?hd y se incrementa el numero de habitaciones visitadas (hab-visitadas) en una unidad.

```
(:durative-action mover-robot-vacio
 :parameters (?r - robot ?ho - habitacion ?hd - habitacion)
 :duration (= ?duration (/ (mover-robot-sin-caja) (velocidad ?r)))
 :condition (and (at start (puerta ?ho ?hd))
                 (at start (esta ?r ?ho))
                 (over all (libre ?r)))
 :effect (and (at start (not (esta ?r ?ho)))
              (at end (esta ?r ?hd))
              (at end (increase (hab-visitadas) 1))))
```

mover-robot-caja:

Los parámetros que necesita la acción son un robot ?r, una caja ?c, una habitación origen ?ho y una habitación destino ?hd.

La acción tiene una duración definida por el tiempo que tarda en moverse un robot con caja de una habitación a otra a través de una puerta partido entre la velocidad a la que trabaja dicho robot.

Como condición al principio se tiene que cumplir que exista una puerta entre la habitación origen ?ho y la habitación destino ?hd y se tiene que cumplir que el robot ?r este en la habitación origen ?ho. Durante toda la acción se tiene que cumplir que el robot ?r este ocupado (con la caja ?c).

Como efecto al principio de la acción se elimina que el robot ?r este en la habitación origen ?ho. Al final de la acción se añade que el robot ?r esta en la habitación destino ?hd y se incrementa el numero de habitaciones visitadas (hab-visitadas) en una unidad.

```
(:durative-action mover-robot-caja
 :parameters (?r - robot ?c - caja ?ho - habitacion ?hd - habitacion)
 :duration (= ?duration (/ (mover-robot-caja) (velocidad ?r)))
 :condition (and (at start (puerta ?ho ?hd))
                 (at start (esta ?r ?ho))
                 (over all (ocupado ?r ?c)))
 :effect (and (at start (not (esta ?r ?ho)))
              (at end (esta ?r ?hd))
              (at end (increase (hab-visitadas) 1))))
```

3. Especificación del problema

Nuestros problemas son escenarios con una configuración de robots, de cajas situadas en habitaciones y de habitaciones adyacentes interconectadas por una configuración dada de puertas.

Como he comentado en la descripción del problema tenemos dos escenarios definidos uno que es trivial que es el escenario1, y otro que es bastante mas complejo que es el escenario2.

En este apartado de la memoria vamos a explicar ambos escenarios.

Escenario 1

Este escenario es muy trivial como he comentado ya que esta compuesto de un robot, una caja y cuatro habitaciones conectadas por tres puertas, este escenario ha sido útil para realizar comprobaciones triviales de que el problema funciona correctamente, vamos a explicar paso a paso el archivo “escenario1.pddl”:

Nombre del problema y dominio al que pertenece:

El nombre del problema (escenario) es escenario1 y pertenece al dominio robots-cajas que hemos definido en el apartado anterior de la memoria.

```
; Nombre del problema
(define (problem escenario1)

; Dominio al que corresponde
(:domain robots-cajas)
```

Objetos definidos:

Este escenario como ya he comentado tiene un robot que hemos llamado robot1, una caja que hemos llamado caja1 y cuatro habitaciones que reciben el nombre de habitacion[1..4].

```
; Objetos definidos en el problema
(:objects
    robot1 - robot
    caja1 - caja
    habitacion1 - habitacion
    habitacion2 - habitacion
    habitacion3 - habitacion
    habitacion4 - habitacion)
```

Estado inicial:

En este punto especificamos los parámetros iniciales con los que va a empezar a trabajar nuestro escenario, este escenario tiene inicialmente el robot1 en la habitacion1, la caja1 en la habitacion4, el robot1 esta libre, es decir, no tiene una caja cargada inicialmente, se definen la conexión entre habitaciones mediante puertas donde hemos de tener en cuenta que si la habitacion1 esta conectada con la habitacion2 también hemos de definir que la habitacion2 esta conectada con la habitacion1. También definimos inicialmente la velocidad del robot1 que es “1”, las habitaciones visitadas que tiene el valor “0” ya que aún no se ha visitado ninguna habitación, el tiempo que se tarda en cargar una caja que es “3”, el tiempo de descargar una caja que es “2”, el tiempo que necesita un robot llevando una caja para moverse de una habitación a otra pasando solo por una puerta que es “2” y por último el tiempo que necesita un robot sin caja para moverse de una habitación a otra pasando solo por una puerta que es “1”.

```
; Parametros iniciales
(:init
    (esta robot1 habitacion1)
    (esta caja1 habitacion4)
    (libre robot1)
    (puerta habitacion1 habitacion2)
    (puerta habitacion2 habitacion1)
    (puerta habitacion2 habitacion3)
    (puerta habitacion3 habitacion2)
    (puerta habitacion3 habitacion4)
    (puerta habitacion4 habitacion3)
    (= (velocidad robot1) 1)
    (= (hab-visitadas) 0)
    (= (tiempo-cargar-caja) 3)
    (= (tiempo-descargar-caja) 2)
    (= (mover-robot-caja) 2)
    (= (mover-robot-sin-caja) 1))
```

Objetivo:

El objetivo de este escenario es que la caja1 acabe en la habitacion3 y que el robot1 acabe en el origen que es la habitacion1.

```
; Objetivo
(:goal (and (esta robot1 habitacion1)
            (esta caja1 habitacion3)))
```

Métrica:

La métrica nos permite determinar que calidad tiene el plan devuelto por el planificador, en nuestro caso la métrica depende de la duración total del plan (total-time) y de el número de habitaciones visitadas por el robot (hab-visitadas).

```
; Funcion a minimizar
(:metric minimize (+ (* 2 (total-time)) (* 0.02 (hab-visitadas))))
```

Escenario 2

Este escenario es mas complejo ya que tiene trece habitaciones, dos robots y tres cajas, con este escenario es con el que vamos a trabajar en las ejecuciones que veremos posteriormente con los planificadores lpg-td y mips-xxl. A continuación vamos a describir el archivo “escenario2.pddl”:

Nombre del problema y dominio al que pertenece:

El nombre del problema (escenario) es escenario2 y pertenece al dominio robots-cajas.

```
; Nombre del problema
(define (problem escenario2)

; Dominio al que corresponde
(:domain robots-cajas)
```

Objetos definidos:

Este escenario tiene mas definiciones de objetos que son dos robots que reciben el nombre de robot1 y robot2, tres cajas con nombre caja1, caja2 y caja3 y tenemos trece habitaciones que reciben el nombre habitacion[1..13].

```
; Objetos definidos en el problema
(:objects
    robot1 - robot
    robot2 - robot
    caja1 - caja
    caja2 - caja
    caja3 - caja
    habitacion1 - habitacion
    habitacion2 - habitacion
    habitacion3 - habitacion
    habitacion4 - habitacion
    habitacion5 - habitacion
    habitacion6 - habitacion
    habitacion7 - habitacion
    habitacion8 - habitacion
    habitacion9 - habitacion
    habitacion10 - habitacion
    habitacion11 - habitacion
    habitacion12 - habitacion
    habitacion13 - habitacion)
```

Estado inicial:

En este punto especificamos los parámetros iniciales con los que va a empezar a trabajar nuestro escenario, este escenario tiene inicialmente el robot1 y el robot2 en la habitacion1, la caja1 en la habitacion8, la caja2 en la habitacion13 y la caja3 en la habitacion6, el robot1 y el robot2 están libres, es decir, no tienen una caja cargada inicialmente, se definen la conexión entre habitaciones mediante puertas que corresponde a la ilustración mostrada en la descripción del problema.
También definimos inicialmente la velocidad del robot1 que es “1” y la velocidad del robot2 que es “2”, las habitaciones visitadas que tiene el valor “0” ya que aún no se ha visitado ninguna habitación, el tiempo que se tarda en cargar una caja que es “3”, el tiempo de descargar una caja que es “2”, el tiempo que necesita un robot llevando una caja para moverse de una habitación a otra pasando solo por una puerta que es “3” y por último el tiempo que necesita un robot sin caja para moverse de una habitación a otra pasando solo por una puerta que es “2”.

```
; Parametros iniciales
(:init
    (esta robot1 habitacion1)
    (esta robot2 habitacion1)
    (esta caja1 habitacion8)
    (esta caja2 habitacion13)
    (esta caja3 habitacion6)
    (libre robot1)
    (libre robot2)
    (puerta habitacion1 habitacion2)
    (puerta habitacion2 habitacion1)
    (puerta habitacion2 habitacion4)
    (puerta habitacion3 habitacion5)
    (puerta habitacion3 habitacion9)
    (puerta habitacion4 habitacion2)
    (puerta habitacion4 habitacion5)
    (puerta habitacion4 habitacion7)
    (puerta habitacion5 habitacion3)
    (puerta habitacion5 habitacion4)
    (puerta habitacion5 habitacion6)
    (puerta habitacion6 habitacion5)
    (puerta habitacion7 habitacion4)
    (puerta habitacion8 habitacion11)
    (puerta habitacion9 habitacion3)
    (puerta habitacion9 habitacion10)
    (puerta habitacion9 habitacion12)
    (puerta habitacion10 habitacion9)
    (puerta habitacion10 habitacion13)
    (puerta habitacion11 habitacion8)
    (puerta habitacion11 habitacion12)
    (puerta habitacion12 habitacion9)
    (puerta habitacion12 habitacion11)
    (puerta habitacion13 habitacion10)
    (= (velocidad robot1) 1)
    (= (velocidad robot2) 2)
    (= (hab-visitadas) 0)
    (= (tiempo-cargar-caja) 3)
    (= (tiempo-descargar-caja) 2)
    (= (mover-robot-caja) 3)
    (= (mover-robot-sin-caja) 2))
```

Objetivo:

El objetivo de este escenario es que la caja1 acabe en la habitacion7, la caja2 acabe en la habitacion8, que la caja3 acabe en la habitacion10 y que el robot1 y el robot2 acaben en el origen que es la habitacion1.

```
; Objetivo
(:goal (and (esta robot1 habitacion1)
            (esta robot2 habitacion1)
            (esta caja1 habitacion7)
            (esta caja2 habitacion8)
            (esta caja3 habitacion10)
            ))
```

Métrica:

La métrica es la misma que en el escenario1 que depende de la duración total del plan (total-time) y de el número de habitaciones visitadas por el robot (hab-visitadas).

```
; Funcion a minimizar
(:metric minimize (+ (* 2 (total-time)) (* 0.02 (hab-visitadas))))
```

4. Ejecuciones con diferentes planificadores

En este apartado de la memoria vamos a encontrar soluciones de planificación para el escenario2 que hemos descrito en el punto anterior, para ello vamos a utilizar dos planificadores que son el lpg-td y el mips-xxl y vamos a contrastar resultados y obtener conclusiones.

lpg-td

Vamos a empezar con el planificador lpg-td que ya conocemos de la primera parte de la práctica, para ello vamos a realizar cinco ejecuciones ya que lpg-td no es determinista como hemos visto anteriormente. Las ejecuciones las vamos a hacer con un valor de -n igual a 4 ya que con cinco no suele acabar la ejecución en un tiempo razonable. Para realizar una ejecución tecleamos lo siguiente:

./lpg-td-1.0 -o dominio.pddl -f escenario2.pddl -n 4

Las cinco ejecuciones las adjunto en la práctica en el directorio ejecuciones/lpgtd/ donde cada carpeta ejecucion[1..5] tiene las cuatro soluciones obtenidas de las cuales nos interesa "plan_escenario2.pddl_4.SOL" que es el mejor plan de esa ejecución, también adjunto una captura de los valores obtenidos que muestro en la tabla comparativa.

Escenario2 con lpg-td				
	T. Ejecución	N.º Acciones	Total-Time	Calidad plan
Ejecución1	0.156s	42	38.00	80.20
Ejecución2	0.428s	42	59.00	122.20
Ejecución3	0.272s	30	47.00	97.00
Ejecución4	0.668s	48	44.00	92.80
Ejecución5	0.408	44	40.00	84.40

Conclusiones de la tabla obtenida:

La mejor solución obtenida es la ejecución1 que se ha obtenido en 0.156s y se basa en un plan de 42 acciones con una duración total del plan de 38 con una calidad del plan de 80.2 según la métrica que hemos comentado antes.

La peor solución obtenida es la ejecución2 que se ha obtenido en 0.428s y se basa en un plan de 42 acciones con una duración total del plan de 59 con una calidad del plan de 122.2.

La solución basándonos en la mediana es la ejecución4 que se ha obtenido en 0.668s y se basa en un plan de 48 acciones con una duración total del plan de 44 con una calidad del plan de 92.8.

El archivo “ plan_escenario2.pddl_4.SOL” nos devuelve el mejor plan obtenido en esa ejecución, vamos a observar que el archivo contiene el plan en la siguiente imagen, por ejemplo el mejor plan obtenido de las cinco ejecuciones lo encontramos en ejecuciones/lpgtd/ejecucion1/plan_escenario2.pddl_4.SOL:

```
0.0003:   (MOVER-ROBOT-VACIO ROBOT1 HABITACION1 HABITACION2) [2.0000]
2.0005:   (MOVER-ROBOT-VACIO ROBOT1 HABITACION2 HABITACION4) [2.0000]
4.0008:   (MOVER-ROBOT-VACIO ROBOT1 HABITACION4 HABITACION5) [2.0000]
6.0010:   (MOVER-ROBOT-VACIO ROBOT1 HABITACION5 HABITACION6) [2.0000]
8.0013:   (COGER-CAJA ROBOT1 CAJA3 HABITACION6) [3.0000]
11.0015:   (MOVER-ROBOT-CAJA ROBOT1 CAJA3 HABITACION6 HABITACION5) [3.0000]
14.0017:   (MOVER-ROBOT-CAJA ROBOT1 CAJA3 HABITACION5 HABITACION3) [3.0000]
17.0020:   (MOVER-ROBOT-CAJA ROBOT1 CAJA3 HABITACION3 HABITACION9) [3.0000]
20.0023:   (MOVER-ROBOT-CAJA ROBOT1 CAJA3 HABITACION9 HABITACION10) [3.0000]
23.0025:   (DEJAR-CAJA ROBOT1 CAJA3 HABITACION10) [2.0000]
25.0028:   (MOVER-ROBOT-VACIO ROBOT1 HABITACION10 HABITACION9) [2.0000]
27.0030:   (MOVER-ROBOT-VACIO ROBOT1 HABITACION9 HABITACION3) [2.0000]
29.0033:   (MOVER-ROBOT-VACIO ROBOT1 HABITACION3 HABITACION5) [2.0000]
31.0035:   (MOVER-ROBOT-VACIO ROBOT1 HABITACION5 HABITACION4) [2.0000]
33.0037:   (MOVER-ROBOT-VACIO ROBOT1 HABITACION4 HABITACION2) [2.0000]
0.0040:   (MOVER-ROBOT-VACIO ROBOT2 HABITACION1 HABITACION2) [1.0000]
1.0043:   (MOVER-ROBOT-VACIO ROBOT2 HABITACION2 HABITACION4) [1.0000]
2.0045:   (MOVER-ROBOT-VACIO ROBOT2 HABITACION4 HABITACION5) [1.0000]
3.0048:   (MOVER-ROBOT-VACIO ROBOT2 HABITACION5 HABITACION3) [1.0000]
4.0050:   (MOVER-ROBOT-VACIO ROBOT2 HABITACION3 HABITACION9) [1.0000]
5.0052:   (MOVER-ROBOT-VACIO ROBOT2 HABITACION9 HABITACION10) [1.0000]
6.0055:   (MOVER-ROBOT-VACIO ROBOT2 HABITACION10 HABITACION13) [1.0000]
7.0058:   (COGER-CAJA ROBOT2 CAJA2 HABITACION13) [3.0000]
10.0060:   (MOVER-ROBOT-CAJA ROBOT2 CAJA2 HABITACION13 HABITACION10) [1.5000]
11.5063:   (MOVER-ROBOT-CAJA ROBOT2 CAJA2 HABITACION10 HABITACION9) [1.5000]
13.0065:   (MOVER-ROBOT-CAJA ROBOT2 CAJA2 HABITACION9 HABITACION12) [1.5000]
14.5068:   (MOVER-ROBOT-CAJA ROBOT2 CAJA2 HABITACION12 HABITACION11) [1.5000]
16.0070:   (MOVER-ROBOT-CAJA ROBOT2 CAJA2 HABITACION11 HABITACION8) [1.5000]
17.5072:   (DEJAR-CAJA ROBOT2 CAJA2 HABITACION8) [2.0000]
19.5075:   (COGER-CAJA ROBOT2 CAJA1 HABITACION8) [3.0000]
22.5077:   (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION8 HABITACION11) [1.5000]
24.0080:   (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION11 HABITACION12) [1.5000]
25.5082:   (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION12 HABITACION9) [1.5000]
27.0085:   (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION9 HABITACION3) [1.5000]
28.5088:   (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION3 HABITACION5) [1.5000]
30.0090:   (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION5 HABITACION4) [1.5000]
31.5093:   (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION4 HABITACION7) [1.5000]
33.0095:   (DEJAR-CAJA ROBOT2 CAJA1 HABITACION7) [2.0000]
35.0098:   (MOVER-ROBOT-VACIO ROBOT2 HABITACION7 HABITACION4) [1.0000]
36.0100:   (MOVER-ROBOT-VACIO ROBOT2 HABITACION4 HABITACION2) [1.0000]
37.0103:   (MOVER-ROBOT-VACIO ROBOT2 HABITACION2 HABITACION1) [1.0000]
35.0105:   (MOVER-ROBOT-VACIO ROBOT1 HABITACION2 HABITACION1) [2.0000]
```

mips-xxl

Ahora vamos a trabajar con el planificador mips-xxl y vamos a ejecutarlo de la siguiente manera:

./mips-xxl -o dominio.pddl -f escenario2.pddl -O

Este planificador si que es determinista por lo tanto todas las ejecuciones devuelven el mismo resultado. El resultado lo encontramos en la carpeta ejecuciones/mips/ejecucion/ donde tenemos cuatro archivos, pero el archivo que nos interesa es el archivo "ffPSolution.soln" ya que paraleliza las acciones obteniendo el valor real de la duración del plan. El resultado lo mostramos en la siguiente tabla:

Escenario2 con mips-xxl				
	T. Ejecución	N.º Acciones	Total-Time	Calidad plan
Ejecución	6.304s	44	62.80	44.00

A continuación mostramos el plan obtenido tras la ejecución que esta en el archivo comentado anteriormente "ffPSolution.soln":

```
0.00: (MOVER-ROBOT-VACIO ROBOT2 HABITACION1 HABITACION2 ) [1.00]
1.01: (MOVER-ROBOT-VACIO ROBOT2 HABITACION2 HABITACION4 ) [1.00]
2.02: (MOVER-ROBOT-VACIO ROBOT2 HABITACION4 HABITACION5 ) [1.00]
2.04: (MOVER-ROBOT-VACIO ROBOT1 HABITACION1 HABITACION2 ) [2.00]
3.03: (MOVER-ROBOT-VACIO ROBOT2 HABITACION5 HABITACION3 ) [1.00]
4.05: (MOVER-ROBOT-VACIO ROBOT1 HABITACION2 HABITACION4 ) [2.00]
6.06: (MOVER-ROBOT-VACIO ROBOT1 HABITACION4 HABITACION5 ) [2.00]
7.07: (MOVER-ROBOT-VACIO ROBOT2 HABITACION3 HABITACION9 ) [1.00]
8.07: (MOVER-ROBOT-VACIO ROBOT1 HABITACION5 HABITACION3 ) [2.00]
9.08: (MOVER-ROBOT-VACIO ROBOT2 HABITACION9 HABITACION10 ) [1.00]
10.09: (MOVER-ROBOT-VACIO ROBOT2 HABITACION10 HABITACION13 ) [1.00]
11.09: (COGER-CAJA ROBOT2 CAJA2 HABITACION13 ) [3.00]
14.09: (MOVER-ROBOT-CAJA ROBOT2 CAJA2 HABITACION13 HABITACION10 ) [1.50]
15.60: (MOVER-ROBOT-CAJA ROBOT2 CAJA2 HABITACION10 HABITACION9 ) [1.50]
17.11: (MOVER-ROBOT-CAJA ROBOT2 CAJA2 HABITACION9 HABITACION12 ) [1.50]
18.62: (MOVER-ROBOT-CAJA ROBOT2 CAJA2 HABITACION12 HABITACION11 ) [1.50]
20.13: (MOVER-ROBOT-CAJA ROBOT2 CAJA2 HABITACION11 HABITACION8 ) [1.50]
21.63: (DEJAR-CAJA ROBOT2 CAJA2 HABITACION8 ) [2.00]
23.64: (COGER-CAJA ROBOT2 CAJA1 HABITACION8 ) [3.00]
26.64: (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION8 HABITACION11 ) [1.50]
27.66: (MOVER-ROBOT-VACIO ROBOT1 HABITACION3 HABITACION5 ) [2.00]
28.15: (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION11 HABITACION12 ) [1.50]
29.66: (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION12 HABITACION9 ) [1.50]
30.68: (MOVER-ROBOT-VACIO ROBOT1 HABITACION5 HABITACION6 ) [2.00]
31.17: (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION9 HABITACION3 ) [1.50]
32.68: (COGER-CAJA ROBOT1 CAJA3 HABITACION6 ) [3.00]
35.68: (MOVER-ROBOT-CAJA ROBOT1 CAJA3 HABITACION6 HABITACION5 ) [3.00]
38.69: (MOVER-ROBOT-CAJA ROBOT1 CAJA3 HABITACION5 HABITACION3 ) [3.00]
41.70: (MOVER-ROBOT-CAJA ROBOT1 CAJA3 HABITACION3 HABITACION9 ) [3.00]
44.71: (MOVER-ROBOT-CAJA ROBOT1 CAJA3 HABITACION9 HABITACION10 ) [3.00]
47.71: (DEJAR-CAJA ROBOT1 CAJA3 HABITACION10 ) [2.00]
49.71: (MOVER-ROBOT-VACIO ROBOT1 HABITACION10 HABITACION9 ) [2.00]
51.72: (MOVER-ROBOT-VACIO ROBOT1 HABITACION9 HABITACION3 ) [2.00]
52.23: (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION3 HABITACION5 ) [1.50]
53.74: (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION5 HABITACION4 ) [1.50]
54.76: (MOVER-ROBOT-VACIO ROBOT1 HABITACION3 HABITACION5 ) [2.00]
55.25: (MOVER-ROBOT-CAJA ROBOT2 CAJA1 HABITACION4 HABITACION7 ) [1.50]
56.75: (DEJAR-CAJA ROBOT2 CAJA1 HABITACION7 ) [2.00]
56.77: (MOVER-ROBOT-VACIO ROBOT1 HABITACION5 HABITACION4 ) [2.00]
58.75: (MOVER-ROBOT-VACIO ROBOT2 HABITACION7 HABITACION4 ) [1.00]
58.78: (MOVER-ROBOT-VACIO ROBOT1 HABITACION4 HABITACION2 ) [2.00]
59.76: (MOVER-ROBOT-VACIO ROBOT2 HABITACION4 HABITACION2 ) [1.00]
60.79: (MOVER-ROBOT-VACIO ROBOT1 HABITACION2 HABITACION1 ) [2.00]
61.80: (MOVER-ROBOT-VACIO ROBOT2 HABITACION2 HABITACION1 ) [1.00]
```

5. Tabla comparativa entre planificadores

Escenario2			
	T. Ejecución	N.º Acciones	Total-Time
Lpg-td	0.668s	48	44.00
Mips-xxl	6.304s	44	62.80

Nota: En el planificador lpg-td hemos cogido el valor de la mediana de las cinco ejecuciones obtenidas porque es el mas realista para realizar la comparación.
Observamos que es mejor el plan obtenido por el planificador lpg-td ya que aunque necesita mas acciones para alcanzar el objetivo (48 frente a 44) la duración del plan es menor (44 frente a 62.80), en tiempo de ejecución también es mejor lpg-td (0.668s frentre a 6.304s).
Por lo tanto para nuestro escenario2 la mejor opción es coger el planificador lpg-td.

CONCLUSIÓN

En la primera parte de esta práctica hemos aprendido a utilizar dos planificadores que se utilizan en la actualidad como son lpg-td y mips-xxl realizando ejecuciones de un conjunto de problemas ya definidos y realizando comparaciones entre ambos planificadores. Hemos visto que ambos utilizan heurísticas pero de diferente forma ya que lpg-td esta basado en búsqueda local utilizando grafos de planificación y mips-xxl utiliza grafos de planificación pero con los efectos delete relajados, esto nos ha servido para asimilar mejor lo visto en teoría. En la segunda parte de la práctica hemos definido un problema de planificación basado en un escenario con robots, cajas y habitaciones del cual partiendo de un estado inicial buscamos un plan de acciones (solución) para obtener un estado final, para ello hemos profundizado en el lenguaje PDDL para definir el dominio del problema y dos escenarios para dicho dominio estudiando como se define un problema en PDDL, una vez definido el problema hemos realizado ejecuciones y comparaciones con los planificadores lpg-td y mips-xxl observando que para nuestro problema es mejor la utilización de lpg-td.

AUTOCRÍTICA

La práctica ha sido muy completa, centrando primero la utilización de los planificadores y después la definición de un problema desde cero en PDDL que es un lenguaje muy usado. Creo que el problema definido es un buen ejemplo para entender que es un verdadero problema de planificación y porque es importante el concepto de heurística para poder resolverlo, lo hemos podido comprender concretamente al buscar un plan para el escenario2 donde tenemos un numero ya considerable de habitaciones y varias cajas y hemos visto que un planificador se centra mas en devolver una solución (un plan para resolver el problema) que una solución óptima (el plan óptimo) ya que esta solución es mucho mas compleja y cara.