Técnicas, Entornos y Aplicaciones de Inteligencia Artificial

Evaluación Práctica-2: Planificación. 2023-2024.

Nombre: César Martínez Chico

MUY IMPORTANTE. NORMAS DEL EXAMEN

1) **Poned el nombre y leed bien cada una de las preguntas.**
2) **Subid a Poliformat los archivos .pddl correspondientes a la práctica ya realizada y los ficheros necesarios para cada uno de los ejercicios. Se puede subir un archivo .zip**
3) **Contestad a las preguntas siguientes de forma razonada, rellenando los huecos con las respuestas. Se deberá subir también este archivo con las respuestas en formato .RTF o .PDF**
4) **Se debe partir de la práctica y realizar todos los cambios necesarios. Los ejercicios y apartados son incrementales. Es decir, el resultado del ejercicio 1 se utiliza como base para el ejercicio 2, el ejercicio 2 se usa como base para el ejercicio 3, etc.**
5) **Los resultados de TODOS los planes que se obtengan con el planificador LPG deberán realizarse con la misma semilla: -seed 100.**
6) **Ejecutad lpg con la opción "-n 3". En caso de que no encuentre la tercera solución, mostrad el último plan resultante (podría ser *-pddl_3.SOL, *-pddl_2.SOL o *-pddl_1.SOL)**
7) **ÚNICAMENTE se admitirán los trabajos subidos a la tarea de Poliformat y dentro del plazo de la tarea.**
8) **No está permitida la utilización de aplicaciones de mensajería.**

Tiempo: 75 minutos.

---

1. **(*2 puntos, Tiempo estimado: 15'*)** Utilizando el dominio y problema realizado en la práctica, añadid la siguiente nueva información:

- Un lugar G. La distancia de cualquier lugar A..F a G es de 60 km. La distancia de G a cualquier lugar A..F es de 70 km.
- G es ZVR y tiene un punto de recarga.
- Hay dos nuevos paquetes: paqueteL4 (ligero) y paqueteP3 (pesado). paqueteL4 se debe transportar de B a G y paqueteP3 de D a F.

Utilizad la siguiente métrica: (:metric minimize (+ (* 0.5 (total-time)) (* 0.7 (coste-recargas))))

Indica las modificaciones realizadas:

He modificado OBJECT: para añadir la zona y los paquetes:

(:objects

dronL1 dronL2 dronL3 - dronLigero

dronP1 dronP2 - dronPesado

paqueteL1 paqueteL2 paqueteL3 paqueteL4 - paqueteLigero

paqueteP1 paqueteP2 paqueteP3- paquetePesado

A B C D E F G - lugar

)

He añadido las siguientes distancias:
(= (distancia A G) 60)
(= (distancia B G) 60)
(= (distancia C G) 60)
(= (distancia D G) 60)
(= (distancia E G) 60)
(= (distancia F G) 60)

(=(distancia G A) 70)

(=(distancia G B) 70)

(=(distancia G C) 70)

(=(distancia G D) 70)

(=(distancia G E) 70)

(=(distancia G F) 70)

(=(distancia G G) 0)

He añadido estos hechos para modelar que G es una ZVR y un punto de recarga, que además estará libre al comienzo:
(ZVR G)
(puntoRecarga G)
(librePuntoRecarga G)

Y para los paquetes he añadido:

(at paqueteL4 B) ; inicialmente en B
(at paqueteP3 D) ; inicialmente en D

Y por ultimo he modificado la goal añadiendo el destino de los nuevos paquetes:

(:goal (and

(at paqueteL1 C)

(at paqueteL2 B)

(at paqueteL3 D)

(at paqueteL4 G)

```
        (at paqueteP1 F)
        (at paqueteP2 E)
        (at paqueteP3 F)
        (at dronL1 B)
        (at dronL2 B)
        (at dronL3 B)
        (at dronP1 E)
        (at dronP2 E)
    )
)
```

Ejecuta lpg y muestra el último plan resultante:

```
; Version LPG-td-1.0
; Seed 100
; Command line: /cygdrive/c/Users/C◆sar/TODO/UNIVERSIDAD/CUARTO/TIA/practica2/lpg-td -seed 100 -o domainDrones1.pddl -f problemaDrones1.pddl -n 3
; Problem problemaDrones1.pddl
; Time 0.75
; Search time 0.75
; Parsing time 0.00
; Mutex time 0.00
; Quality 115.00
```

Time 0.75

0.0003: (RECARGARBATERIA DRONL2 A) [9.0000]

0.0005: (RECARGARBATERIA DRONL3 C) [10.0000]

9.0007: (RECARGARBATERIA DRONL1 A) [8.0000]

17.0010: (RECOGERPAQUETEENDRONLIGERO PAQUETEL2 DRONL1 A) [1.0000]

10.0013: (VOLARDRONLIGERO C F DRONL3) [8.0000]

18.0015: (RECOGERPAQUETEENDRONLIGERO PAQUETEL3 DRONL3 F) [1.0000]

19.0018: (VOLARDRONLIGERO F D DRONL3) [10.0000]

29.0020: (ENTREGARPAQUETE PAQUETEL3 DRONL3 D) [2.0000]

18.0023: (VOLARDRONLIGERO A B DRONL1) [8.0000]

26.0025: (ENTREGARPAQUETE PAQUETEL2 DRONL1 B) [2.0000]

9.0028: (VOLARDRONLIGERO A C DRONL2) [5.0000]

14.0030: (VOLARDRONLIGERO C E DRONL2) [14.0000]

28.0033: (VOLARDRONLIGERO E B DRONL2) [6.0000]

34.0035: (RECARGARBATERIA DRONL2 B) [9.0000]

43.0037: (RECOGERPAQUETEENDRONLIGERO PAQUETEL4 DRONL2 B) [1.0000]

44.0040: (VOLARDRONLIGERO B G DRONL2) [15.0000]

59.0042: (ENTREGARPAQUETE PAQUETEL4 DRONL2 G) [2.0000]

61.0045: (RECARGARBATERIA DRONL2 G) [9.0000]

70.0048: (VOLARDRONLIGERO G B DRONL2) [17.5000]

0.0050: (RECOGERPAQUETEENDRONPESADO PAQUETEP3 DRONP2 D) [2.0000]

2.0052: (ENTREGARPAQUETE PAQUETEP3 DRONP2 D) [2.0000]

4.0055: (RECOGERPAQUETEENDRONPESADO PAQUETEP1 DRONP2 D) [2.0000]

6.0058: (VOLARDRONPESADO D F DRONP2) [20.0000]

26.0060: (ENTREGARPAQUETE PAQUETEP1 DRONP2 F) [2.0000]

31.0063: (VOLARDRONLIGERO D C DRONL3) [8.0000]

39.0065: (VOLARDRONLIGERO C B DRONL3) [10.0000]

28.0068: (RECOGERPAQUETEENDRONLIGERO PAQUETEL1 DRONL1 B) [1.0000]

29.0070: (VOLARDRONLIGERO B A DRONL1) [8.0000]

37.0073: (VOLARDRONLIGERO A C DRONL1) [10.0000]

47.0075: (ENTREGARPAQUETE PAQUETEL1 DRONL1 C) [2.0000]

49.0078: (VOLARDRONLIGERO C B DRONL1) [20.0000]

28.0080: (RECARGARBATERIA DRONP2 F) [25.0000]

53.0083: (VOLARDRONPESADO F D DRONP2) [20.0000]

73.0085: (RECOGERPAQUETEENDRONPESADO PAQUETEP3 DRONP2 D) [2.0000]

75.0088: (VOLARDRONPESADO D F DRONP2) [20.0000]

95.0090: (ENTREGARPAQUETE PAQUETEP3 DRONP2 F) [2.0000]

97.0092: (RECOGERPAQUETEENDRONPESADO PAQUETEP2 DRONP2 F) [2.0000]

99.0095: (VOLARDRONPESADO F D DRONP2) [20.0000]

119.0098: (VOLARDRONPESADO D E DRONP2) [21.0000]

140.0100: (ENTREGARPAQUETE PAQUETEP2 DRONP2 E) [2.0000]

2. ***(4 puntos, Tiempo estimado: 30')*** El control sobre los drones se ha vuelto más estricto. Por lo tanto, **al menos** cada dos acciones de vuelo (podría ser también después de cada vuelo), todo dron debe pasar una operación de mantenimiento obligatoria, que implica un nuevo coste. Hasta que no se haya finalizado dicho mantenimiento, el dron no podrá volver a volar. Una vez realizado el mantenimiento, el dron ya podrá volver a volar hasta dos veces más. Actualmente, las tareas de mantenimiento se realizan en los mismos lugares donde hay un punto de recarga, aunque se espera habilitar otros lugares distintos, por lo que hay que dejar preparado el modelo. Lógicamente, durante la tarea de mantenimiento un dron no puede tener ningún paquete.
Existen dos formas de llevar a cabo el mantenimiento:

- Mantenimiento de un único dron (solo se puede procesar un dron en cada momento). Esta acción tiene una duración de 5 unidades y añade un coste de mantenimiento de 5 unidades.
- Mantenimiento de dos drones distintos (solo se pueden procesar dos drones en cada momento). Esta acción tiene una duración de 8 unidades y añade un coste de mantenimiento de 9 unidades. NOTA: se recomienda añadir una condición de este tipo: (over all (not (= ?dron1 ?dron2))) para asegurar que los dos drones sean distintos. "not" ya está definida en PDDL, por lo que no hay que realizar ninguna definición adicional.

En el mismo lugar y en el mismo momento, las dos acciones de mantenimiento son excluyentes; es decir, se puede ejecutar la primera o la segunda, pero no las dos a la vez.

En el estado inicial todos los drones están recién revisados, por lo que podrían realizar hasta dos vuelos. Asimismo, se debe contemplar la siguiente métrica:

(:metric minimize (+ (coste-mantenimiento) (+ (* 0.5 (total-time)) (* 0.7 (coste-recargas)))))

Indica las modificaciones realizadas:

He añadido estas funciones:

```
(duracionRepararUno)
(duracionRepararDos) ;2
(coste-mantenimiento)
(vuelos ?dron - dron)
```

Y los inicializado así:

```
(=(coste-mantenimiento) 0)
(= (duracionRepararUno) 5)
(= (duracionRepararDos) 8)
(= (vuelos dron) 0)
```

Este ultimo para todos los drones

Los predicados:

```
(reparaciones ?zona -lugar)
```

```
(reparando ?zona - lugar)
```

Para evitar que se reparen mas de uno (o dos, segun)
Inicializadas:

```
(reparaciones A)
(reparaciones B)
(reparaciones C)
(reparaciones F)
(reparaciones G)
```

Y he añadido las acciones:

```
(:durative-action repararUnoSolo
 :parameters (?dron - dron  ?zona - lugar)
 :duration   (= ?duration (duracionRepararUno))
 :condition  (and (over all (reparaciones ?zona)) ;durante todo el rato tiene que
ser zona de reparar
                 (at start (not(reparando ?zona))) ;zona libre
                 (at start (libre ?dron)) ;que el dron no este haciendo otra
cosa, como recargar
                 (at start(< (vuelos ?dron) 2 ))
                 (over all (not(llevandoPaquete ?dron))) ;no puedo llevar un
paquete
                 (over all (at ?dron ?zona)) ;el dron ha de estar en la zona todo
el tiempo
             )
 :effect     (and (at start (reparando ?zona)) ; no esta libre hasta terminar
                 (at start (not(libre ?dron)))
                 (at end(assign (vuelos ?dron) 2 ) ) ;cambiamos los vuelos del
dron.
                 (at end(increase (coste-mantenimiento) 5) ) ;funcion a minimizar
                 (at end (libre ?dron))
                 (at end (not(reparando ?zona))) ;hemos terminado de recargar,
liberamos el punto.
             )
)
```

```
(:durative-action repararDos
 :parameters (?dron1 - dron ?dron2 - dron  ?zona - lugar)
 :duration   (= ?duration (duracionRepararDos))
 :condition  (and (over all (reparaciones ?zona)) ;durante todo el rato tiene que
ser zona de reparar
                  (at start (not(reparando ?zona))) ;zona libre
                  (at start (libre ?dron1)) ;que el dron no este haciendo otra
cosa, como recargar
                  (at start (libre ?dron2))
                  (at start(< (vuelos ?dron1) 2 ))
                  (at start(< (vuelos ?dron2) 2 ))
                  (over all (not(llevandoPaquete ?dron1))) ;no puedo llevar un
paquete
                  (over all (not(llevandoPaquete ?dron2)))
                  (over all (not (= ?dron1 ?dron2)))
                  (over all (at ?dron1 ?zona)) ;el dron ha de estar en la zona
todo el tiempo
                  (over all (at ?dron2 ?zona))
             )
 :effect     (and (at start (reparando ?zona)) ; no esta libre hasta terminar
                  (at start (not(libre ?dron1)))
                  (at start (not(libre ?dron2)))
                  (at end(assign (vuelos ?dron1) 2 ) ) ;cambiamos los vuelos del
dron.
                  (at end(assign (vuelos ?dron2) 2 ) ) ;cambiamos los vuelos del
dron.
                  (at end(increase (coste-mantenimiento) 9) ) ;funcion a minimizar
                  (at end (libre ?dron1))
                  (at end (libre ?dron2))
                  (at end (not(reparando ?zona))) ;hemos terminado de recargar,
liberamos el punto.
             )
)
```

Son parecidas simplemente cambiando que las comprobaciones que se hacen para un dron, se hagan para los dos, que no sean el mismo dron y cambiando los costes.

Y además he cambiado que las acciones de volar tengan en cuenta que hay que tener vuelos disponibles y luego decrementalros después

```
(:durative-action volarDronLigero
 :parameters (?zona1 - lugar ?zona2 - lugar ?dron - dronLigero )
 :duration   ( = ?duration (/ (distancia ?zona1 ?zona2) (velocidad ?dron) ) )
 :condition  (and (at start (at ?dron ?zona1))
                  (at start(> (vuelos ?dron) 0) )
                  (at start(>= (bateria ?dron) (distancia ?zona1 ?zona2)));al
inicio la bateria del dron ha de ser mayor que la distancia recorrida.
             )
 :effect     (and (at start(not(at ?dron ?zona1) ) ) ;dron ya no en zona 1
                  (at end(at ?dron ?zona2)) ;dron zona 2
                  (at end(decrease (bateria ?dron) (distancia ?zona1 ?zona2) ) )
;reducimos la bateria
```

```
                (at end(decrease (vuelos ?dron) 1 ) ) ;le quitamos un vuelo
            )
)

(:durative-action volarDronPesado
 :parameters (?zona1 - lugar ?zona2 - lugar ?dron - dronPesado )
 :duration   ( = ?duration (/ (distancia ?zona1 ?zona2) (velocidad ?dron) ) )
 :condition  (and (at start (at ?dron ?zona1)) ;dron en la zona
                (at start(>= (bateria ?dron) (distancia ?zona1 ?zona2))) ;al
inicio la bateria del dron ha de ser mayor que la distancia recorrida.
                (over all (not(ZVR ?zona2))) ;comprobar que zona2 no sea ZVR
                (at start(> (vuelos ?dron) 0 ))
            ) ;lo demas analogo a volarligero
 :effect     (and (at start(not(at ?dron ?zona1) ) ) ;dron ya no en zona 1
                (at end(at ?dron ?zona2)) ; dron en zona 2
                (at end(decrease (bateria ?dron) (distancia ?zona1 ?zona2) ) )
;reducimos la bateria.
                 (at end(decrease (vuelos ?dron) 1 ) ) ;le quitamos un vuelo
            )
)
```

Ejecuta lpg y muestra el último plan resultante:

No me encuentra un plan

3. ***(2.5 puntos, Tiempo estimado: 20')*** La acción de recoger un paquete pesado requiere de un brazo robótico especial. Todos los lugares disponen de uno de estos brazos, pero cada brazo solo puede utilizarse como máximo 2 veces por problema.

Indica las modificaciones realizadas:

Añadire la function

```
(brazo ?zona - lugar )
```

Y las inicializare:

```
(=(brazo A) 2)
(=(brazo B) 2)
(=(brazo C) 2)
(=(brazo D) 2)
(=(brazo E) 2)
```

```
(=(brazo F) 2)
(=(brazo G) 2)
```

Y no me da tiempo a mas PERO MIRA EL ULTIMO EJERCICIO

Ejecuta lpg y muestra el último plan resultante:

¿Qué pasaría si exigimos que el brazo robótico situado en D se tenga que usar como máximo 1 vez? Indica los cambios necesarios y el resultado obtenido.

4. ***(1.5 puntos, Tiempo estimado: 10')*** ¿Qué cambios serían necesarios para que la duración de la acción de volar dependa de si el dron transporta un paquete o no? Es decir, volar vacío tardaría menos tiempo que volar con un paquete cargado. ¿Qué cambios adicionales habría que realizar si volar con un paquete cargado depende también del tipo de paquete (ligero/pesado)? Explica razonadamente todos los cambios necesarios. Puedes explicarlo de palabra o implementarlo en PDDL.

A) Bastaria con modificar la duration de la acción de volar, multiplicando su valor por si lleva o no lleva paquete. Si por ejemplo volar con paquete durase 3 mas, seria:
:duration (= ?duration (+ (/ (distancia ?zona1 ?zona2) (velocidad ?dron)) llevarPaqueteTiempo))

Y definiríamos una funcion llevarPaqueteTiempo que inicilizariamos con el valor que deseemos que tenga,

Para el tipo de paquete habría que hacer lo mismo pero definir una funcion para cada tiempo con cada tipo de paquete

:duration (= ?duration (+ (/ (distancia ?zona1 ?zona2) (velocidad ?dron)) llevarPaqueteTiempoLigero))
:duration (= ?duration (+ (/ (distancia ?zona1 ?zona2) (velocidad ?dron)) llevarPaqueteTiempoPesado))