Técnicas de los sistemas inteligentes (2014-2015)
Grado en Ingeniería Informática
Universidad de Granada

Práctica 3.1 - Planificación

Ignacio Martín Requena

29 de junio de 2015

Índice

Índice de figuras

1. Ejercicio 1

1.1. Enunciado

Escribir un dominio de planificación en PDDL para que un planificador pueda encontrar planes de actuación para uno o varios robots como soluciones a problemas de distribución de paquetes entre habitaciones. En el material de esta sesión de prácticas hay un fichero ejemplo de un problema para este tipo de dominio, que se corresponde con el estado inicial y objetivo en la figura de más abajo. Probar que el dominio escrito genera plan para distintos problemas (definidos por el alumno) en los que varíe la cantidad de robots,la cantidad de paquetes a distribuir y la cantidad de habitaciones conectadas.

1.2. Solución

Para la resolución de este problema se ha implementado:

- **Elementos**: tres elementos: un objeto, un paquete, un robot y una habitacion.
- **Predicados**: para representar si un objeto está en una habitación (at), si una habitación está conectada con otra (conectada), si el robot no posee nada (vacio) y si el robot esta llevando un paquete (llevando).
- **Acciones**: se han definido tres acciones:
  - **Acción mover:** necesaria para representar el movimiento llevado a cabo por el robot para desplazar un paquete de una habitacion a otra.
  - **Acción soltar:** que define el hecho de que el robot puede soltar ?obj en la habitación ?h
  - **Acción coger:** para definir la acion de coger un paquete de una habitación por parte del robot

En general el procedimiento seguido para el desarrollo de este problema ha sido el de pensar cuales eran las acciones necesarias para realizar el plan completo de mover un objeto de una habitación conectada a otra.

1.3. Comprobación de la solución

El problema utilizado para comprobar la solución de que el dominio implementado es correcto ha sido:

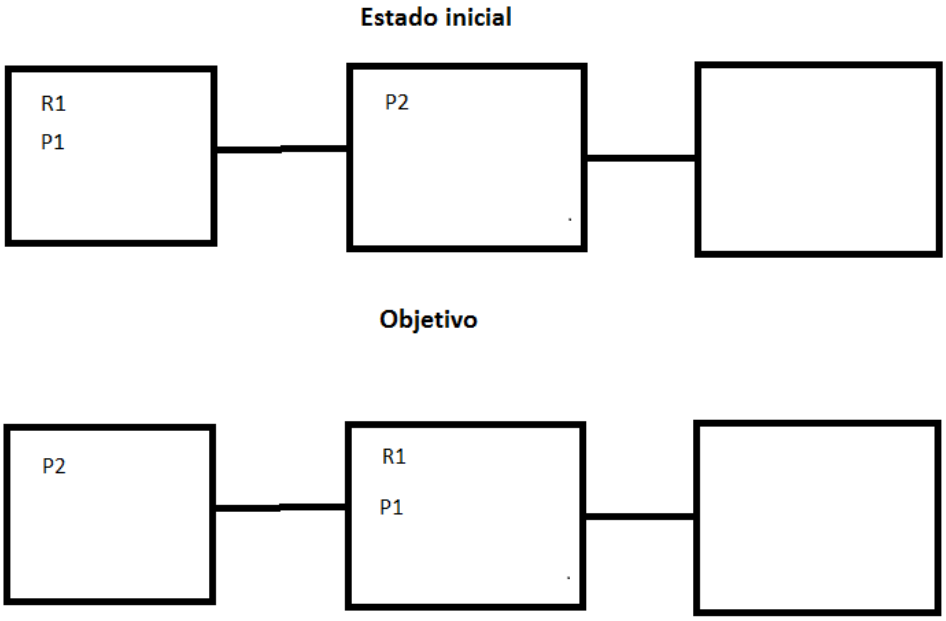

Figura 1.1: Representación estado inicial y objetivo problema 1 dominio 1

```
(define (problem RDistribuye-1)
(:domain RobotDistribuidor)
(:objects
        r1 - robot
        p1 - paquete
        p2 - paquete

        hab0 - habitacion
        hab1 - habitacion
        hab2 - habitacion

        )
(:init
        (at r1 hab0)
        (at p1 hab0)
        (at p2 hab2)

```

```
18          (vacio r1)
19
20          (conectada hab0 hab1)
21          (conectada hab1 hab0)
22          (conectada hab2 hab1)
23          (conectada hab1 hab2)
24  )
25  (:goal (and
26          (at r1 hab1)
27          (at p2 hab0)
28          (at p1 hab2)
29          ))
30  )
```

He decidido escoger una solución con un robot, dos paquetes y tres habitaciones. Otra modificación respecto al fichero original proporcionado como ejemplo ha sido la de incluir en el estado inicial el hecho de que el robot no posee nada al principio (sin esta condición inicial el plan falla).

2. Ejercicio 2

2.1. Enunciado

Escribir un dominio de planificación en PDDL, modificando el dominio del anterior ejercicio, de tal manera que se tenga en cuenta que la acción de moverse de una habitación a otra consume una cantidad de batería y, por tanto, requiere que el robot tenga nivel de batería para moverse. Además, considerar que hay una nueva acción de carga de batería que permite reponer la batería. Considerar para ello que se ha definido un predicado (cambio n1 n2–nivelbat) que representa un cambio en el nivel de batería desde un nivel n1 a un nivel n2. En el material de esta sesión de prácticas hay un fichero ejemplo de un problema para este tipo de dominio.

2.2. Solución

Para la resolución de este problema se ha partido de la solución proporcionada en el ejercicio 1 realizando algunas modificaciones. En concreto, las modificaciones han sido:

- **Elementos**: Se ha añadido un nuevo tipo de objeto para representar la existencia de la funte de bateria (fuente)
- **Predicados**: Se han añadido dos nuevos predicados para representar le cambio de la batería y el nivel de esta.
- **Acciones**:
    - **Acción mover:** En la acción mover se han añadido dos nuevos parámetros de tipo fuente, la batería antes y después. También se han modificado las precondiciones y los efectos de esta para representar el desgaste de la batería.

En general el procedimiento seguido para el desarrollo de este problema ha sido el de determinar cuando el nivel de batería debería bajar (en la acción de mover) asi como que estados de batería son necesarios representar (el de antes y el de despues de la carga).

Lo que influye en la cantidad y niveles de batería definidos en el estado inicial es, basicamente, la colocación de las baterías y a que bateria tiene acceso cada robot.

2.3. Comprobación de la solución

El problema utilizado para comprobar la solución de que el dominio implementado es correcto ha sido:

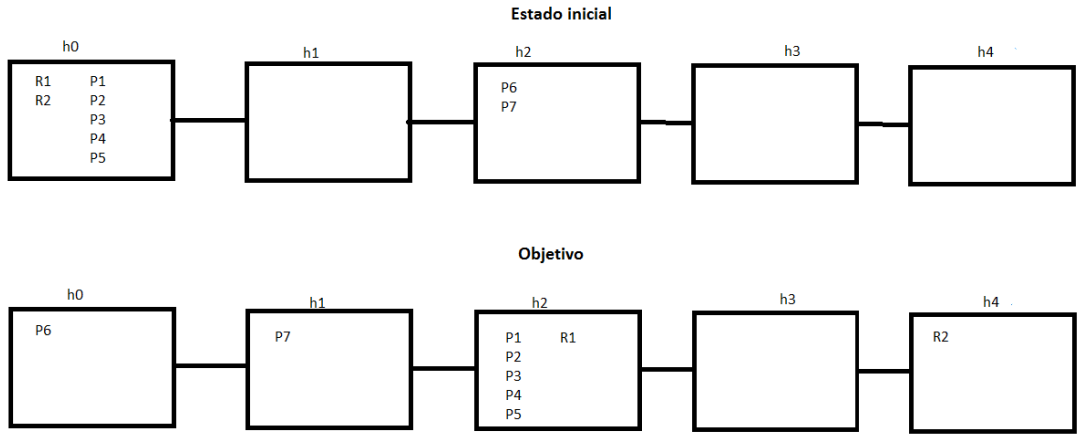

Figura 2.1: Representación estado inicial y objetivo problema 1 dominio 2

```
(define (problem RDistribuye-1)
        (:domain RobotDistribuidor)
        (:objects
                r1 r2 - robot
                p1 p2 p3 p4 p5 p6 p7 - paquete
                hab0 hab1 hab2 hab3 hab4 - habitacion
                f0 f1 f2 - fuente
        )
        (:init
                (at r1 hab0)
                (at r2 hab0)
                (at p1 hab0)
                (at p2 hab0)
                (at p3 hab0)
                (at p4 hab0)
                (at p5 hab0)
                (at p6 hab2)
```

```
                    (at p7 hab2)

                    (vacio r1)
                    (vacio r2)

                    (nivelbateria r1 f2)
                    (nivelbateria r2 f2)

                    (cambio f2 f1)
                    (cambio f1 f0)
                    (cambio f0 f2)

                    (conectada hab0 hab1)
                    (conectada hab1 hab0)
                    (conectada hab2 hab1)
                    (conectada hab1 hab2)
                    (conectada hab3 hab2)
                    (conectada hab2 hab3)
                    (conectada hab3 hab4)
                    (conectada hab4 hab3)
          )
          (:goal (and
                    (at p6 hab0)
                    (at p7 hab1)
                    (at p1 hab2)
                    (at p2 hab2)
                    (at p3 hab2)
                    (at p4 hab2)
                    (at p5 hab2)
                    (at r1 hab2)
                    (at r2 hab4)
                    )
          )

)
```

En este caso se ha optado por elegir un problema con dos robots, siete paquetes, cuatro habitaciones y 3 fuentes. Las habitaciones conetadas son: Habitacion 0 con 1, habitación 1 con 2, habitación 2 con 3 y habitación 3 con 4. A su vez las fuentes se pueden cambiar de la forma: f2 a f1, f1 a f0 y f0 a f2.

3. Ejercicio 3

3.1. Enunciado

Escribir un dominio de planificación en PDDL, modificando el dominio del anterior ejercicio, de manera que se puedan utilizar ahora dos acciones diferentes, moverse rápido y moverse lento tales que moverse rápido consume más unidades de fuel que moverse lento. Probarlo con varios problemas.

3.2. Solución

Para la resolución de este problema se ha tomado como base para la implementación la solución adoptada en el problema anterior. Los principales cambios con respecto a este han sido:

- **Predicados**: Se ha añadido un nuevo predicado para representar el cambio en el nivel de la batería si el movimiento de nuestro robot es rápido.
- **Acciones**: Se ha añadido una nueva acción:
  - **Acción mover rápido:** Esta acción tiene como objetivo simular el cambio en el nivel de batería de forma más rápida que si el robot se moviera a velocidad normal. Para ello se ha seguido el mismo esquema que para la accion mover a velocidad normal pero a la hora de representarel cambio en el nivel de la batería se ha usado el nuevo predicado cambio-rapido.

En general el procedimiento seguido para el desarrollo de este problema ha sido el de pensar qué añadido tenía el hecho de que la velocidad del robot condicionara el nivel de batería llegando a la conclusión de que únicamente añadiendo un nuevo predicado para representar este hecho era suficiente.

3.3. Comprobación de la solución

Los problemas utilizados para comprobar la solución de que el dominio implementado es correcto han sido:

- **Problema 1:**

```

(define (problem RDistribuye-1)
        (:domain RobotDistribuidor)
        (:objects
        r1 r2 - robot
        p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 - paquete
        hab0 hab1 hab2 hab3 hab4 - habitacion
        fl1 fl2 fl0 - fuente
        )
```

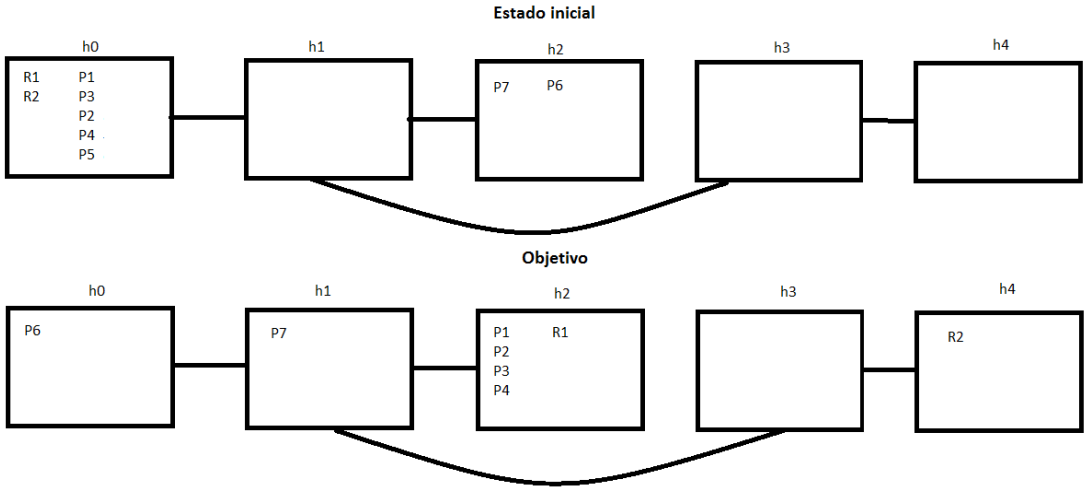

Figura 3.1: Representación estado inicial y objetivo problema 1 dominio 3

```
10          (:init
11          (at r1 hab0)
12          (at r2 hab0)
13
14          (at p1 hab0)
15          (at p3 hab0)
16          (at p2 hab0)
17          (at p4 hab0)
18          (at p5 hab0)
19          (at p7 hab2)
20          (at p6 hab2)
21
22          (vacio r1)
23          (vacio r2)
24
25          (nivelbateria r1 fl2)
26          (nivelbateria r2 fl2)
27
28
29          (cambio fl2 fl1)
30          (cambio fl1 fl0)
31          (cambio fl0 fl2)
32
33          (cambio-rapido fl2 fl0)
34          (cambio-rapido fl0 fl2)
35
36          (conectada hab1 hab0)
37          (conectada hab0 hab1)
```

```
38          (conectada hab1 hab2)
39          (conectada hab2 hab1)
40          (conectada hab3 hab1)
41          (conectada hab1 hab3)
42          (conectada hab4 hab3)
43          (conectada hab3 hab4)
44          )
45
46          (:goal (and
47          (at p1 hab2)
48          (at p3 hab2)
49          (at p4 hab2)
50          (at p2 hab2)
51          (at p5 hab2)
52          (at p6 hab0)
53          (at p7 hab0)
54
55          (at r1 hab2)
56          (at r2 hab4)
57          ))
58
59  )
```

En este caso se ha optado por elegir un problema con dos robots, siete paquetes, cuatro habitaciones y 3 fuentes. Las habitaciones conetadas son: Habitacion 0 con 1, habitación 1 con 2, habitación 2 con 3 y habitación 3 con 4. A su vez las fuentes se pueden cambiar de la forma: f2 a f1, f1 a f0 y f0 a f2.

- **Problema 2**:

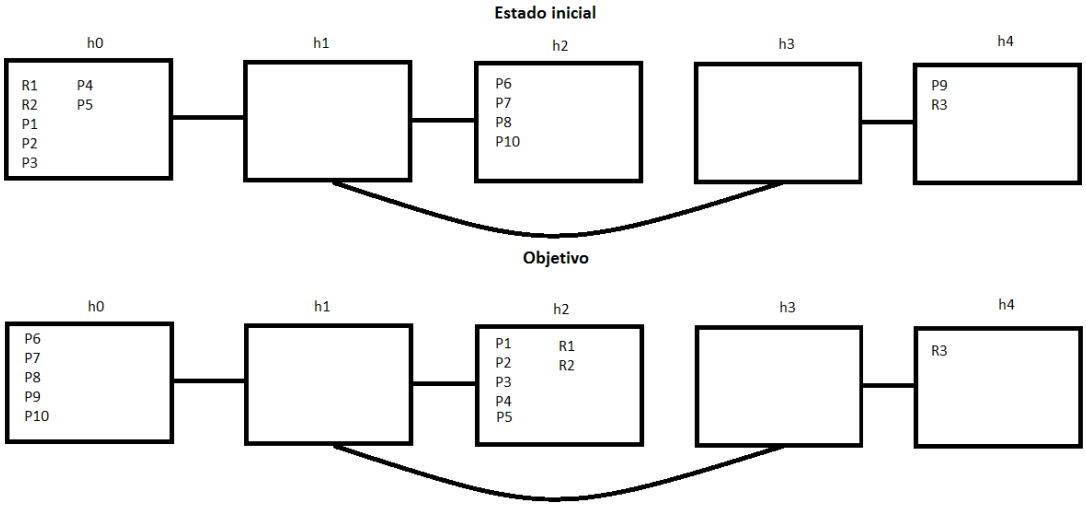

Figura 3.2: Representación estado inicial y objetivo problema 2 dominio 3

```
(define (problem RDistribuye-1)
(:domain RobotDistribuidor)
(:objects
        r1 r2 r3 - robot
        p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 - paquete
        hab0 hab1 hab2 hab3 hab4 - habitacion
        fl1 fl2 fl3 - fuente
        )
(:init
        (at r1 hab0)
        (at r2 hab0)
        (at r3 hab4)

        (at p1 hab0)
        (at p2 hab0)
        (at p3 hab0)
        (at p4 hab0)
        (at p5 hab0)
        (at p6 hab2)
        (at p7 hab2)
        (at p8 hab2)
        (at p9 hab4)
        (at p10 hab2)

        (vacio r1)
        (vacio r2)
        (vacio r3)
```

```
28
29          (nivelbateria r1 fl2)
30          (nivelbateria r2 fl2)
31          (nivelbateria r3 fl2)
32
33          (cambio fl3 fl2)
34          (cambio fl2 fl1)
35          (cambio fl1 fl2)
36
37          (cambio-rapido fl2 fl3)
38          (cambio-rapido fl1 fl2)
39
40          (conectada hab0 hab1)
41          (conectada hab1 hab0)
42          (conectada hab2 hab1)
43          (conectada hab1 hab2)
44          (conectada hab1 hab3)
45          (conectada hab3 hab1)
46          (conectada hab3 hab4)
47          (conectada hab4 hab3)
48  )
49  (:goal (and
50          (at p1 hab2)
51          (at p2 hab2)
52          (at p3 hab2)
53          (at p4 hab2)
54          (at p6 hab0)
55          (at p5 hab2)
56          (at p7 hab0)
57          (at p8 hab0)
58          (at p9 hab0)
59          (at p10 hab0)
60
61          (at r1 hab2)
62          (at r2 hab2)
63          (at r3 hab4)
64          ))
65
66  )
```

En este segundo problema implementado se ha intentado complicar el estado inicial. Este estado está compuesto por tres robots, dos de ellos en la habitación 0 y uno en la 4; cuatro habitaciones, diez paquetes y tres fuentes de batería.

4. Ejercicio 4

4.1. Enunciado

Escribir un dominio de planificación en PDDL 2.1 que responda a los requisitos explicados en los anteriores ejercicios, utilizando las capacidades expresivas de PDDL 2.1, es decir, representando funciones numéricas. Al igual que en los anteriores ejercicios, definir distintos problemas para comprobar que vuestro dominio es correcto.

4.2. Solución

Para la resolución de este problema se ha modificado el dominio del problema 3 para representar la carga y descarga del robot a través de funciones numéricas en lugar de a través de predicados. En concreto los elementos añadidos y modificados con respecto al problema anterior han sido:

- **Predicados**: Se han eliminado los predicados que representaban el cambio de batería del robot así como la carga y descarga. Esto es porque la representación de este estado se ha realizado mediante funciones.
- **Funciones**: Se ha añadido una función (descarga) que representa el agotamiento de la batería
- **Acciones**: Se han modificado y/o añadido las acciones:
  - **Acción mover y mover rápido:** Se han modificado estas dos acciones con una llamada a la función descarga que, en función de si la acción es mover o mover rápido el nivel de la batería se decrementa en 5 o en 10 respectivamente.
  - **Acción cargar**: Esta nueva acción representa el momento en el que el robot se carga. Como precondición he definido que el nivel de batería sea menor a 5 y, una vez cargado el robot, este nivel aumenta hasta 50 (es decir, se carga completamente)

En general el procedimiento seguido para el desarrollo de este problema ha sido el de sustituir los anteriores objetos y predicados del problema que representaban el estado de la batería por una función numérica que representa la misma situación. A partir de esto se han modificado y añadido las acciones pertinentes para el control de la misma.

4.3. Comprobación de la solución

Los problemas utilizados para comprobar la solución de que el dominio implementado es correcto han sido:

- **Problema 1:**

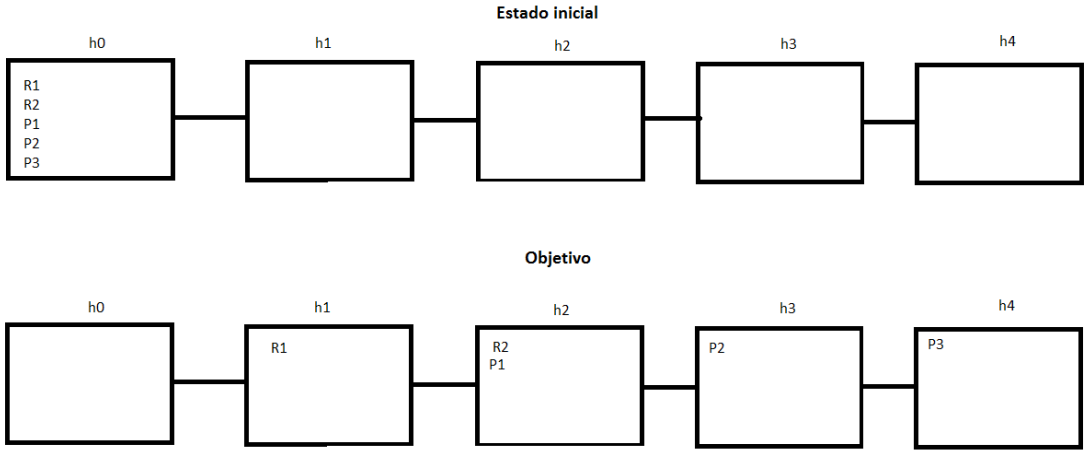

Figura 4.1: Representación estado inicial y objetivo problema 1 dominio 4

```
(define (problem RDistribuye-1)
(:domain RobotDistribuidor)
(:objects
        r1 r2 - robot
        p1 p2 p3 p4 p5 p6 p7 - paquete
        hab0 hab1 hab2 hab3 hab4 - habitacion
        )
(:init
        (at r1 hab0)
        (at r2 hab0)
        (at p1 hab0)
        (at p2 hab0)
        (at p3 hab0)

        (vacio r1)
        (vacio r2)

        (= (descarga r1 ) 50)
        (= (descarga r2 ) 50)

        (conectada hab0 hab1)
        (conectada hab1 hab0)
```

```
23          (conectada hab2 hab1)
24          (conectada hab1 hab2)
25          (conectada hab2 hab3)
26          (conectada hab3 hab2)
27          (conectada hab3 hab4)
28          (conectada hab4 hab3)
29  )
30  (:goal (and
31          (at p1 hab2)
32          (at p2 hab3)
33          (at p3 hab4)
34
35          (at r1 hab1)
36          (at r2 hab2)
37          ))
38
39  )
```

En este dominio se han usado 2 robots, 7 paquetes y 5 habitaciones.

- **Problema 2:**

  En este segundo problema implementado se ha añadido un nuevo robot posicionado junto a un único paquete con el fin de comprobar si PDDL resuelve de forma eficiente el problema, es decir, utiliza el robot 3 para mover el paquete 9 y no hace ir a otro robot a por el. A demás se han definido 10 paquetes y 5 habitaciones.

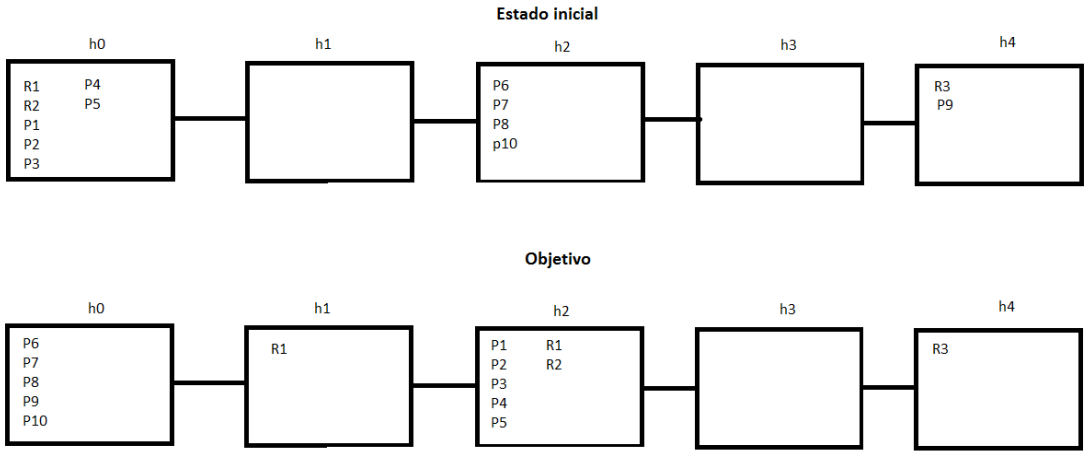

Figura 4.2: Representación estado inicial y objetivo problema 2 dominio 4

```
(define (problem RDistribuye-1)
(:domain RobotDistribuidor)
(:objects
        r1 r2 r3 - robot
        p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 - paquete
        hab0 hab1 hab2 hab3 hab4 - habitacion
        )
(:init
        (at r1 hab0)
        (at r2 hab0)
        (at r3 hab4)
        (at p1 hab0)
        (at p2 hab0)
        (at p3 hab0)
        (at p4 hab0)
        (at p5 hab0)
        (at p6 hab2)
        (at p7 hab2)
        (at p8 hab2)
        (at p9 hab4)
        (at p10 hab2)

        (vacio r1)
        (vacio r2)
        (vacio r3)

        (= (descarga r1 ) 50)
        (= (descarga r2 ) 50)
        (= (descarga r3 ) 50)

        (conectada hab0 hab1)
        (conectada hab1 hab0)
        (conectada hab2 hab1)
        (conectada hab1 hab2)
        (conectada hab1 hab3)
        (conectada hab3 hab1)
        (conectada hab3 hab4)
        (conectada hab4 hab3)
)
(:goal (and
        (at p6 hab0)
        (at p7 hab0)
        (at p8 hab0)
        (at p9 hab0)
        (at p10 hab0)
        (at p1 hab2)
        (at p2 hab2)
        (at p3 hab2)
        (at p4 hab2)
```

```
50          (at p5 hab2)
51
52          (at r1 hab2)
53          (at r2 hab2)
54          (at r3 hab4)
55          ))
56
57  )
```

- **Problema 3:**

  Este tercer y último problema ha sido una variación del representado en la **Figura 4.2**. En concreto se ha añadido un robot mas y se ha cambiado el estado objetivo para complicar aún más el problema (aun así el planificador Metric-FF solo ha necesitado 54 pasos para resolverlo).

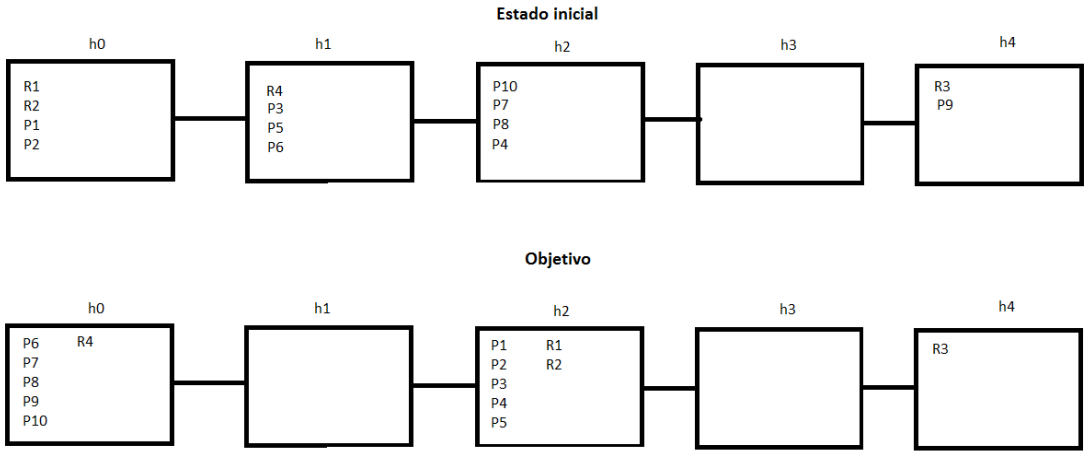

Figura 4.3: Representación estado inicial y objetivo problema 3 dominio 4

```
1
2   (define (problem RDistribuye-1)
3   (:domain RobotDistribuidor)
4   (:objects
5           r1 r2 r3 r4 - robot
6           p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 - paquete
7           hab0 hab1 hab2 hab3 hab4 - habitacion
8           )
9   (:init
10          (at r1 hab0)
11          (at r2 hab0)
12          (at r3 hab4)
```

```
          (at r4 hab1)

          (at p1 hab0)
          (at p2 hab0)
          (at p3 hab1)
          (at p4 hab2)
          (at p5 hab1)
          (at p6 hab1)
          (at p7 hab2)
          (at p8 hab2)
          (at p9 hab4)
          (at p10 hab2)

          (vacio r1)
          (vacio r2)
          (vacio r3)
          (vacio r4)


          (= (descarga r1 ) 50)
          (= (descarga r2 ) 50)
          (= (descarga r3 ) 50)
          (= (descarga r4 ) 50)


          (conectada hab0 hab1)
          (conectada hab1 hab0)
          (conectada hab2 hab1)
          (conectada hab1 hab2)
          (conectada hab1 hab3)
          (conectada hab3 hab1)
          (conectada hab3 hab4)
          (conectada hab4 hab3)
)
(:goal (and
          (at p6 hab0)
          (at p7 hab0)
          (at p8 hab0)
          (at p9 hab0)
          (at p10 hab0)
          (at p1 hab2)
          (at p2 hab2)
          (at p3 hab2)
          (at p4 hab2)
          (at p5 hab2)

          (at r1 hab2)
          (at r2 hab2)
          (at r3 hab4)
```

```
62            (at r4 hab0)
63            ))
64
65  )
```