UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MIARFID
Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital
Universitat Politècnica de València

Resolución del problema de dominio de Robots mediante técnicas de planificación

Planificación Inteligente

Autor: Jacobo López Fernández
Juan Antonio López Ramírez

Curso 2019-2020

Índice general

Índice general 1

1 Introducción al problema 3

2 Dominio Proposicional 5

2.1 Tipos y predicados . . . 5
2.2 Acciones . . . 6
2.2.1 1. Mover robot . . . 6
2.2.2 2. Dar taza . . . 6
2.2.3 3. Llenar taza . . . 7
2.2.4 4. Servir taza . . . 7
2.2.5 5. Coger taza . . . 7
2.3 Instancia del problema . . . 8
2.4 Resultados obtenidos . . . 9
2.4.1 FF . . . 9
2.4.2 LPG . . . 10
2.4.3 LPG con timesteps . . . 11
2.4.4 Optic . . . 11
2.5 Instancias alternativas . . . 11

3 Dominio Temporal 21

3.1 Acciones en el dominio temporal . . . 21
3.2 Nueva instancia del problema . . . 23
3.3 Resultados obtenidos . . . 24
3.4 Instancias alternativas . . . 25
3.4.1 Resultados para 2 robots y 3 usuarios . . . 26
3.4.2 Resultados para 1 robot y 3 usuarios . . . 27
3.4.3 Resultados para 3 robots y 3 usuarios . . . 28
3.5 Conclusiones . . . 30

4 Dominio con recursos numéricos 31

4.1 Batería . . . 31
4.2 Nuevas acciones con el recurso numérico . . . 31
4.3 Instancias optimizando el gasto de batería y el tiempo . . . 32
4.4 Comparación de resultados. Conclusiones . . . 33
4.4.1 Plan para el recurso no renovable, minimizando la batería y con 2 robots . . . 33
4.4.2 Plan para el recurso no renovable, minimizando la batería y con 3 robots . . . 34
4.4.3 Plan para el recurso renovable, minimizando la batería y con 2 robots 34
4.4.4 Plan para el recurso renovable, minimizando la batería y con 3 robots 35
4.4.5 Plan para el recurso no renovable, minimizando el tiempo y con 2 robots . . . 36
4.4.6 Plan para el recurso no renovable, minimizando el tiempo y con 3 robots . . . 36
4.4.7 Plan para el recurso renovable, minimizando el tiempo y con 2 robots 38
4.4.8 Plan para el recurso renovable, minimizando el tiempo y con 3 robots 38

5 Árbol POP **41**
5.1 Plan inicial 41
5.2 Primera iteración 42
5.3 Segunda iteración 42
5.4 Tercera iteración 43
6 Graphplan **45**
6.1 Plan inicial 45
6.2 Traza del algoritmo 46

CAPÍTULO 1
Introducción al problema

El dominio "Robots" se desarrolla en el escenario de la figura 1.1.

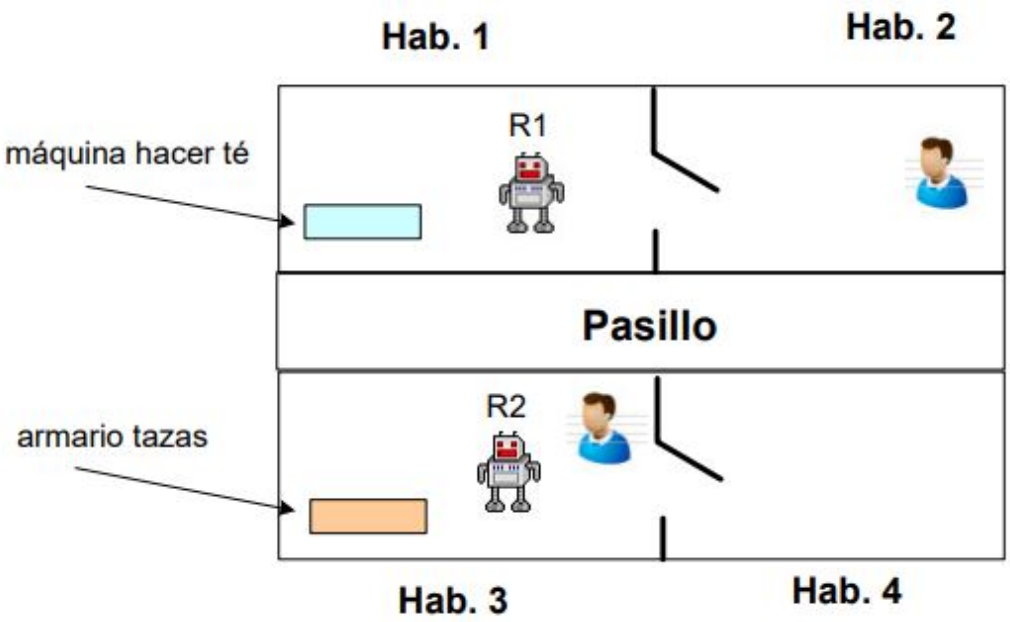

Figura 1.1: Escenario del dominio de Robots.

Hay cuatro habitaciones, todas ellas conectadas con un pasillo central. Además la habitación 1 está conectada con la habitación 2, y la habitación 3 con la habitación 4. En la habitación 1 hay una máquina para hacer té, y en la habitación 3 hay un armario donde se guardan las tazas. Se tienen dos robots, que tienen dos brazos cada uno, y que inicialmente están en la habitación 1 y habitación 3, respectivamente. El primer robot solo puede acceder a la habitación 1, habitación 2 y al pasillo, y el segundo solo puede moverse por la habitación 3, habitación 4 y el pasillo.

El objetivo del problema es satisfacer las peticiones de té de usuarios que se encuentren en las habitaciones. Al principio, partiremos de que hay dos personas que están en las habitaciones 2 y 3 respectivamente, y que desean tomar una taza de té.

CAPÍTULO 2
Dominio Proposicional

Para definir el dominio de nuestro problema, hemos tenido que definir los tipos, los predicados y las acciones. El resultado de todos estos pasos ha sido guardado en el fichero *dominio_robots.pddl*.

2.1 Tipos y predicados

Se han definido 7 tipos distintos, que representan los distintos componentes del problema. Para ello, se ha incluido en nuestro archivo de dominio la siguiente línea de código:

```
(:types robot taza persona lugar armario maquina brazo)
```

Se ha nombrado cada tipo de esa forma para hacer más sencillo el entendimiento del dominio. Por ejemplo, el tipo *lugar* representa cada una de las habitaciones y el pasillo.

Posteriormente, se han definido los predicados que representen el entorno del problema, que son:

```
(:predicates (at ?x - (either persona robot armario maquina) ?y - lugar)
             (on ?x - taza ?y - (either persona armario))
             (have ?x - taza ?y - robot ?z - brazo)
             (linked ?x - lugar ?y - lugar)
             (allowed ?x - robot ?y - lugar)
             (empty ?x - taza)
             (full ?x - taza)
             (ocupado ?x - brazo ?y - robot)
             (libre ?x - brazo ?y - robot))
```

El predicado AT representa que una persona, un robot, el armario de las tazas o la máquina de hacer té están en una determinada habitación. Por ejemplo, (at r1 h1) indica que el robot r1 está en la habitación h1.

ON indica que, o bien una taza está en el armario, o bien la tiene una persona. Por ejemplo, (on t1 a) indica que la taza t1 está en el armario, si previamente hemos definido 'a' de tipo *armario*.

HAVE representa el hecho de que un robot tenga una taza en uno de sus brazos. Este predicado se diferencia del ON en que cuando un robot tiene una taza es necesario explicitar en qué brazo la tiene. Por ejemplo, (have t1 r1 b2) indica que el robot r1 sostiene la taza t1 en su brazo b2.

LINKED indica que dos habitaciones están conectadas. Por ejemplo, (linked l1 l2) indica que las habitaciones l1 y l2 son contiguas y se puede ir de l1 a l2. Es necesario recalcar esto último porque para que se pueda ir de l2 a l1 hay que establecerlo mediante (linked l2 l1).

ALLOWED representa el hecho de que un robot pueda estar en una determinada habitación. Por defecto, en ausencia de un (allowed r1 l1), el robot r1 no podrá acceder a la habitación l1.

EMPTY indica que una taza está vacía, mientras que FULL indica que está llena. Por defecto, ambos hechos son excluyentes para una misma taza.

OCUPADO indica que un brazo de un robot está sosteniendo una taza, mientras que LIBRE representa que un brazo de un robot no está sosteniendo nada. Por defecto, ambos hechos son excluyentes para un mismo brazo de un robot.

2.2 Acciones

Hemos definido un total de 5 acciones.

2.2.1. 1. Mover robot

```
(:action move-robot
       :parameters (?r - robot  ?o - lugar ?d - lugar)
       :precondition (and (at ?r ?o) (linked ?o ?d) (allowed ?r ?d))
           :effect
                 (and (not (at ?r ?o))(at ?r ?d)))
```

Para esta acción, es necesario especificar de qué robot se trata (r), su habitación de procedencia (o) y la habitación a la que queremos que se mueva (d). De ahí, los tres parámetros que hemos puesto.

Las precondiciones indican que el robot r está en su habitación de origen o; que se puede acceder a la habitación destino d desde la de origen o y que el robot r tenga permitido acceder a la habitación destino d.

Por último, los efectos son que r ya no se encuentra en la habitación o y que se encuentra en la habitación d.

2.2.2. 2. Dar taza

```
(:action give-cup
       :parameters (?r1 - robot ?r2 - robot ?t - taza ?l - lugar ?b1 - brazo
           ?b2 - brazo)
       :precondition (and (at ?r1 ?l) (at ?r2 ?l) (have ?t ?r1 ?b1) (libre ?b2
           ?r2) (ocupado ?b1 ?r1))
       :effect
       (and (have ?t ?r2 ?b2) (not (have ?t ?r1 ?b1))(not (ocupado ?b1 ?r1))
           (ocupado ?b2 ?r2)(not (libre ?b2 ?r2))(libre ?b1 ?r1)))
```

En esta acción, los parámetros de entrada son el robot que va a entregar la taza (r1), el robot que la va a recibir (r2), la taza que se va a entregar (t), la habitación en la que se encuentran (l), el brazo del robot que tiene la taza y el brazo del robot que recibirá la taza.

Las precondiciones que se han de cumplir son que los robots r1 y r2 estén en la misma habitación l, que el robot r1 tenga la taza t en uno de sus brazos (este brazo lo hemos denominado b1), que este brazo b1 esté ocupado, y que uno de los brazos del robot r2 esté libre (este brazo lo hemos denominado b2).

Los efectos son que el robot r2 tenga la taza t en su brazo b2, que el robot r1 ya no tenga la taza t en su brazo b1, que este brazo b1 ya no esté ocupado y esté libre, y que el brazo b2 del robot r2 ya no esté libre y ahora esté ocupado.

2.2.3. 3. Llenar taza

```
(:action fill-cup
       :parameters (?r - robot ?m - maquina ?l - lugar ?t - taza ?b - brazo)
       :precondition (and (at ?r ?l)(at ?m ?l)(have ?t ?r ?b)(empty ?t))
       :effect
       (and (not (empty ?t))(full ?t)))
```

Los parámetros son el robot que va a llenar la taza (r), el brazo con el que lo va a hacer (b), la taza que va a llenar (t), la máquina de hacer el té (m) y la habitación (l).

Las precondiciones son que el robot r y que la máquina m estén en la habitación l, que el brazo b del robot r tenga la taza t y que esta esté vacía.

Los efectos indican que la taza t ya no está vacía y que ahora está llena.

2.2.4. 4. Servir taza

```
(:action serve-cup
       :parameters (?r - robot ?p - persona ?l - lugar ?t - taza ?b - brazo)
       :precondition (and (at ?r ?l) (at ?p ?l) (have ?t ?r ?b) (full ?t)
          (ocupado ?b ?r))
       :effect
       (and (not (have ?t ?r ?b))(on ?t ?p)(not (ocupado ?b ?r))(libre ?b ?r)))
```

Los parámetros son el robot que servirá la taza (r), la persona a la que se la entregará (p), la taza a entregar (t), la habitación en la que se encuentran (l) y el brazo del robot que tiene la taza (b).

Las precondiciones son que el robot r y la persona p estén en la misma habitación l, que el robot r tenga la taza t en el brazo b y que la taza t esté vacía.

Los efectos son que el robot r ya no tenga la taza t en su brazo b, que la persona p tenga la taza t, que el brazo b del robot r ya no esté ocupado y que ahora esté libre.

2.2.5. 5. Coger taza

```
(:action get-cup
       :parameters (?r - robot ?a - armario ?l - lugar ?t - taza ?b - brazo)
       :precondition (and (at ?r ?l)(at ?a ?l)(on ?t ?a)(libre ?b ?r))
       :effect
       (and (not (on ?t ?a))(have ?t ?r ?b)(ocupado ?b ?r)(not (libre ?b ?r))))
```

Los parámetros son el robot que va a coger la taza (r), esta misma taza (t), el armario (a), la habitación (l) y el brazo del robot que cogerá la taza (b).

Las precondiciones son que la taza t esté en el armario a, que el armario a y el robot r estén en la misma habitación l y que el brazo b del robot r esté libre.

Los efectos son que la taza t ya no está en el armario a, que ahora la taza t está en el brazo b del robot r, que el brazo b del robot r está ocupado y que ahora el brazo b del robot r ya no está libre.

2.3 Instancia del problema

La instancia del problema la hemos guardado en un fichero llamado *problema_robots.pddl*, en el que añadiremos lo que vamos a explicar a continuación.

Según el escenario que hemos definido previamente, tenemos 4 habitaciones y un pasillo, un armario de tazas, una máquina de hacer té, 2 robots y 2 usuarios que solicitan una taza de té. Cada robot tiene dos brazos, por lo que creamos dos objetos de tipo *brazo*, así aunque los dos robots compartan el nombre de los brazos, se podrá saber a qué robot pertenecen gracias a los predicados LIBRE y OCUPADO. Por otra parte, hemos definido únicamente dos tazas, ya que son solo dos los usuarios que van a solicitar té.

Esto lo implementamos con el siguiente código:

```
(:objects r1 r2 - robot
          t1 t2 - taza
          p1 p2 - persona
          l1 l2 l3 l4 lp - lugar
          a1 - armario
          m1 - maquina
          b1 b2 - brazo
    )
```

Un robot y la máquina de té están en la habitación 1 mientras que el otro robot y el armario de tazas están en la habitación 3. Un usuario está en la habitación 2 y el otro en la 3. Las habitaciones contiguas son la 1 con la 2 y la 3 con la 4, y todas dan al pasillo. El robot de la habitación 1 solo tiene permitido estar en esa habitación, en la 2 y el pasillo, mientras que el robot de la habitación 3 solo puede estar en esa, en la 4 y en el pasillo. Inicialmente, las dos tazas están vacías en el armario y los brazos de los robots están todos libres.

Implementando esto en PDDL obtenemos:

```
(:init
        (at r1 l1)
        (at r2 l3)
        (at p1 l2)
        (at p2 l3)
        (at m1 l1)
        (at a1 l3)

        (on t1 a1)
        (on t2 a1)

        (linked l1 l2)
        (linked l2 l1)
        (linked l1 lp)
```

```
        (linked lp l1)
        (linked l2 lp)
        (linked lp l2)
        (linked l3 l4)
        (linked l4 l3)
        (linked l3 lp)
        (linked lp l3)
        (linked l4 lp)
        (linked lp l4)

        (allowed r1 l1)
        (allowed r1 l2)
        (allowed r1 lp)
        (allowed r2 l3)
        (allowed r2 l4)
        (allowed r2 lp)

        (empty t1)
        (empty t2)

        (libre b1 r1)
        (libre b2 r1)
        (libre b1 r2)
        (libre b2 r2)
    )
```

Podemos apreciar que hemos duplicado los LINKED debido a que, como habíamos comentado antes, hay que explicitar la bidireccionalidad de las habitaciones.

Por último, hay que añadir el objetivo que buscamos alcanzar. En nuestro caso, es que cada usuario tenga una taza llena. Para eso, añadimos el *goal* al código:

```
(:goal (and (on t1 p1) (on t2 p2) (full t1) (full t2)))
```

2.4 Resultados obtenidos

Para el dominio y la instancia del problema recién implementados, hemos ejecutado tres planificadores de cuatro formas distintas: FF, LPG, LPG con timesteps y Optic.

2.4.1. FF

Para usar el planificador FF, ejecutamos la siguiente línea de código:

```
./ff -o dominio_robots.pddl -f problema_robots.pddl -s 0
```

El plan obtenido, en 18 pasos, ha sido:

```
ff: found legal plan as follows
step    0: GET-CUP R2 A1 L3 T2 B1
        1: GET-CUP R2 A1 L3 T1 B2
        2: MOVE-ROBOT R2 L3 LP
        3: MOVE-ROBOT R1 L1 LP
        4: GIVE-CUP R2 R1 T1 LP B2 B1
        5: MOVE-ROBOT R1 LP L1
        6: FILL-CUP R1 M1 L1 T1 B1
        7: MOVE-ROBOT R1 L1 LP
```

```
10          8: MOVE-ROBOT R1 LP L2
11          9: SERVE-CUP R1 P1 L2 T1 B1
12         10: MOVE-ROBOT R1 L2 LP
13         11: GIVE-CUP R2 R1 T2 LP B1 B1
14         12: MOVE-ROBOT R1 LP L1
15         13: FILL-CUP R1 M1 L1 T2 B1
16         14: MOVE-ROBOT R1 L1 LP
17         15: GIVE-CUP R1 R2 T2 LP B1 B1
18         16: MOVE-ROBOT R2 LP L3
19         17: SERVE-CUP R2 P2 L3 T2 B1
```

Como podemos observar, con ese plan alcanzamos el objetivo del problema, aunque no hayan acciones realizadas de forma concurrente.

2.4.2. LPG

Para utilizar el planificador LPG, ejecutamos la siguiente línea de código:

```
1 ./lpg -o dominio_robots.pddl -f problema_robots.pddl -n 1 -out solucion_LPG.txt
```

El plan obtenido, en 17 pasos, ha sido:

```
1  ; Time 0.08
2  ; Search time 0.08
3  ; Parsing time 0.00
4  ; Mutex time 0.00
5  ; NrActions 22
6
7  0:   (GET-CUP R2 A1 L3 T1 B1) [1]
8  0:   (MOVE-ROBOT R1 L1 LP) [1]
9  1:   (MOVE-ROBOT R2 L3 L4) [1]
10 2:   (MOVE-ROBOT R2 L4 LP) [1]
11 3:   (GIVE-CUP R2 R1 T1 LP B1 B2) [1]
12 4:   (MOVE-ROBOT R1 LP L1) [1]
13 4:   (MOVE-ROBOT R2 LP L4) [1]
14 5:   (FILL-CUP R1 M1 L1 T1 B2) [1]
15 5:   (MOVE-ROBOT R2 L4 L3) [1]
16 6:   (MOVE-ROBOT R1 L1 L2) [1]
17 6:   (GET-CUP R2 A1 L3 T2 B1) [1]
18 7:   (SERVE-CUP R1 P1 L2 T1 B2) [1]
19 7:   (MOVE-ROBOT R2 L3 LP) [1]
20 8:   (MOVE-ROBOT R1 L2 LP) [1]
21 9:   (GIVE-CUP R2 R1 T2 LP B1 B2) [1]
22 10:   (MOVE-ROBOT R1 LP L1) [1]
23 11:   (FILL-CUP R1 M1 L1 T2 B2) [1]
24 12:   (MOVE-ROBOT R1 L1 LP) [1]
25 13:   (GIVE-CUP R1 R2 T2 LP B2 B1) [1]
26 14:   (MOVE-ROBOT R2 LP L4) [1]
27 15:   (MOVE-ROBOT R2 L4 L3) [1]
28 16:   (SERVE-CUP R2 P2 L3 T2 B1) [1]
```

Como podemos apreciar, el plan ejecuta 22 acciones, pero alguna de ellas las hace de forma concurrente para minimizar costes, como por ejemplo que el robot r2 coja una taza mientras el robot r1 se dirige al pasillo.

2.4.3. LPG con timesteps

Cuando ejecutamos LPG con timesteps, obtenemos, en el mejor de los casos, 17 pasos por plan (aunque le pongamos un número de timesteps más bajo), que es el mismo obtenido sin timesteps. Esto se debe a que el planificador no es capaz de encontrar una solución con un número de pasos inferior.

2.4.4. Optic

Para ejecutar el planificador de Optic, ejecutamos la siguiente línea de código:

```
./optic-clp dominio_robots.pddl problema_robots.pddl
```

El plan obtenido, en 15 pasos, ha sido:

```
; Plan found with metric 0.014
; States evaluated so far: 43
; States pruned based on pre-heuristic cost lower bound: 0
; Time 0.04
0.000: (get-cup r2 a1 l3 t1 b2) [0.001]
0.000: (get-cup r2 a1 l3 t2 b1) [0.001]
0.000: (move-robot r1 l1 lp) [0.001]
0.001: (move-robot r2 l3 lp) [0.001]
0.002: (give-cup r2 r1 t1 lp b2 b1) [0.001]
0.003: (move-robot r1 lp l1) [0.001]
0.004: (fill-cup r1 m1 l1 t1 b1) [0.001]
0.005: (move-robot r1 l1 l2) [0.001]
0.006: (serve-cup r1 p1 l2 t1 b1) [0.001]
0.007: (move-robot r1 l2 lp) [0.001]
0.008: (give-cup r2 r1 t2 lp b1 b2) [0.001]
0.009: (move-robot r1 lp l1) [0.001]
0.010: (fill-cup r1 m1 l1 t2 b2) [0.001]
0.011: (move-robot r1 l1 lp) [0.001]
0.012: (give-cup r1 r2 t2 lp b2 b1) [0.001]
0.013: (move-robot r2 lp l3) [0.001]
0.014: (serve-cup r2 p2 l3 t2 b1) [0.001]

 * All goal deadlines now no later than 0.014
```

Como podemos observar, el planificador alcanza el objetivo del problema y realizando algunas acciones de forma concurrente, obteniendo un plan que tiene el menor número de pasos vistos hasta ahora, que son 15, por lo que podemos afirmar que el mejor planificador, para esta instancia de este problema, es Optic.

2.5 Instancias alternativas

Hemos probado a modificar la instancia del problema para comprobar como afecta a los planes obtenidos. Es necesario comentar que todas las modificaciones parten de la anterior y solo la primera parte de la original.

La primera modificación ha consistido en cambiar los 2 usuarios que solicitaban té por 5. Por tanto, hemos tenido que añadir 3 tazas más a las 2 previas, además de especificar que están todas vacías y en el armario. Los resultados son planes que alcanzan el objetivo del problema y han sido:

- En FF, un plan de 47 pasos:

```
    ff: found legal plan as follows
step     0: GET-CUP R2 A1 L3 T2 B1
         1: GET-CUP R2 A1 L3 T3 B2
         2: MOVE-ROBOT R2 L3 LP
         3: MOVE-ROBOT R1 L1 LP
         4: GIVE-CUP R2 R1 T2 LP B1 B2
         5: GIVE-CUP R2 R2 T3 LP B2 B1
         6: GIVE-CUP R2 R1 T3 LP B1 B1
         7: MOVE-ROBOT R1 LP L1
         8: FILL-CUP R1 M1 L1 T2 B2
         9: SERVE-CUP R1 P2 L1 T2 B2
        10: MOVE-ROBOT R1 L1 LP
        11: GIVE-CUP R1 R2 T3 LP B1 B1
        12: MOVE-ROBOT R2 LP L3
        13: GET-CUP R2 A1 L3 T1 B2
        14: MOVE-ROBOT R2 L3 LP
        15: GIVE-CUP R2 R1 T3 LP B1 B2
        16: GIVE-CUP R2 R1 T1 LP B2 B1
        17: MOVE-ROBOT R1 LP L1
        18: FILL-CUP R1 M1 L1 T3 B2
        19: FILL-CUP R1 M1 L1 T1 B1
        20: MOVE-ROBOT R1 L1 L2
        21: SERVE-CUP R1 P1 L2 T1 B1
        22: MOVE-ROBOT R1 L2 LP
        23: GIVE-CUP R1 R2 T3 LP B2 B1
        24: MOVE-ROBOT R2 LP L4
        25: SERVE-CUP R2 P3 L4 T3 B1
        26: MOVE-ROBOT R2 L4 L3
        27: GET-CUP R2 A1 L3 T4 B1
        28: MOVE-ROBOT R2 L3 LP
        29: GIVE-CUP R2 R1 T4 LP B1 B2
        30: MOVE-ROBOT R1 LP L1
        31: FILL-CUP R1 M1 L1 T4 B2
        32: MOVE-ROBOT R1 L1 LP
        33: GIVE-CUP R1 R2 T4 LP B2 B1
        34: MOVE-ROBOT R2 LP L4
        35: SERVE-CUP R2 P4 L4 T4 B1
        36: MOVE-ROBOT R2 L4 LP
        37: MOVE-ROBOT R2 LP L3
        38: GET-CUP R2 A1 L3 T5 B1
        39: MOVE-ROBOT R2 L3 LP
        40: GIVE-CUP R2 R1 T5 LP B1 B1
        41: MOVE-ROBOT R1 LP L1
        42: FILL-CUP R1 M1 L1 T5 B1
        43: MOVE-ROBOT R1 L1 LP
        44: GIVE-CUP R1 R2 T5 LP B1 B1
        45: MOVE-ROBOT R2 LP L3
        46: SERVE-CUP R2 P5 L3 T5 B1
```

- En LPG, un plan de 45 pasos:

```
    ; Time 0.11
; Search time 0.08
; Parsing time 0.03
; Mutex time 0.00
; NrActions 52

0:    (GET-CUP R2 A1 L3 T1 B1) [1]
0:    (MOVE-ROBOT R1 L1 LP) [1]
1:    (MOVE-ROBOT R2 L3 LP) [1]
2:    (MOVE-ROBOT R2 LP L4) [1]
3:    (MOVE-ROBOT R2 L4 LP) [1]
```

```
4:   (GIVE-CUP R2 R1 T1 LP B1 B2) [1]
5:   (MOVE-ROBOT R1 LP L2) [1]
5:   (MOVE-ROBOT R2 LP L3) [1]
6:   (MOVE-ROBOT R1 L2 L1) [1]
6:   (GET-CUP R2 A1 L3 T3 B1) [1]
7:   (FILL-CUP R1 M1 L1 T1 B2) [1]
7:   (MOVE-ROBOT R2 L3 LP) [1]
8:   (MOVE-ROBOT R1 L1 L2) [1]
9:   (SERVE-CUP R1 P1 L2 T1 B2) [1]
10:   (MOVE-ROBOT R1 L2 LP) [1]
11:   (GIVE-CUP R2 R1 T3 LP B1 B2) [1]
12:   (MOVE-ROBOT R1 LP L1) [1]
13:   (FILL-CUP R1 M1 L1 T3 B2) [1]
14:   (MOVE-ROBOT R1 L1 LP) [1]
15:   (GIVE-CUP R1 R2 T3 LP B2 B1) [1]
16:   (MOVE-ROBOT R2 LP L4) [1]
17:   (SERVE-CUP R2 P3 L4 T3 B1) [1]
18:   (MOVE-ROBOT R2 L4 LP) [1]
19:   (MOVE-ROBOT R2 LP L3) [1]
20:   (GET-CUP R2 A1 L3 T5 B1) [1]
21:   (MOVE-ROBOT R2 L3 LP) [1]
22:   (GIVE-CUP R2 R1 T5 LP B1 B2) [1]
23:   (MOVE-ROBOT R1 LP L1) [1]
24:   (FILL-CUP R1 M1 L1 T5 B2) [1]
25:   (MOVE-ROBOT R1 L1 LP) [1]
26:   (GIVE-CUP R1 R2 T5 LP B2 B1) [1]
27:   (MOVE-ROBOT R2 LP L4) [1]
28:   (MOVE-ROBOT R2 L4 L3) [1]
29:   (SERVE-CUP R2 P5 L3 T5 B1) [1]
30:   (GET-CUP R2 A1 L3 T2 B1) [1]
31:   (MOVE-ROBOT R2 L3 L4) [1]
32:   (MOVE-ROBOT R2 L4 LP) [1]
33:   (GIVE-CUP R2 R1 T2 LP B1 B2) [1]
34:   (MOVE-ROBOT R1 LP L1) [1]
34:   (MOVE-ROBOT R2 LP L3) [1]
35:   (FILL-CUP R1 M1 L1 T2 B2) [1]
35:   (GET-CUP R2 A1 L3 T4 B1) [1]
36:   (MOVE-ROBOT R2 L3 LP) [1]
36:   (SERVE-CUP R1 P2 L1 T2 B2) [1]
37:   (MOVE-ROBOT R1 L1 LP) [1]
38:   (GIVE-CUP R2 R1 T4 LP B1 B2) [1]
39:   (MOVE-ROBOT R1 LP L1) [1]
40:   (FILL-CUP R1 M1 L1 T4 B2) [1]
41:   (MOVE-ROBOT R1 L1 LP) [1]
42:   (GIVE-CUP R1 R2 T4 LP B2 B1) [1]
43:   (MOVE-ROBOT R2 LP L4) [1]
44:   (SERVE-CUP R2 P4 L4 T4 B1) [1]
```

- En Optic, un plan de 34 pasos:

```
    ; Plan found with metric 0.033
; States evaluated so far: 182
; States pruned based on pre-heuristic cost lower bound: 0
; Time 0.36
0.000: (get-cup r2 a1 l3 t2 b2)  [0.001]
0.000: (get-cup r2 a1 l3 t1 b1)  [0.001]
0.000: (move-robot r1 l1 lp)  [0.001]
0.001: (move-robot r2 l3 lp)  [0.001]
0.002: (give-cup r2 r1 t2 lp b2 b1)  [0.001]
0.002: (give-cup r2 r1 t1 lp b1 b2)  [0.001]
0.003: (move-robot r1 lp l1)  [0.001]
0.003: (move-robot r2 lp l4)  [0.001]
0.004: (fill-cup r1 m1 l1 t1 b2)  [0.001]
0.004: (fill-cup r1 m1 l1 t2 b1)  [0.001]
```

```
0.004: (move-robot r2 l4 l3)  [0.001]
0.005: (serve-cup r1 p2 l1 t2 b1)  [0.001]
0.005: (get-cup r2 a1 l3 t5 b2)  [0.001]
0.005: (get-cup r2 a1 l3 t4 b1)  [0.001]
0.006: (move-robot r1 l1 l2)  [0.001]
0.006: (move-robot r2 l3 lp)  [0.001]
0.007: (serve-cup r1 p1 l2 t1 b2)  [0.001]
0.008: (move-robot r1 l2 lp)  [0.001]
0.009: (give-cup r2 r1 t4 lp b1 b2)  [0.001]
0.010: (move-robot r1 lp l1)  [0.001]
0.011: (fill-cup r1 m1 l1 t4 b2)  [0.001]
0.012: (move-robot r1 l1 lp)  [0.001]
0.013: (give-cup r1 r2 t4 lp b2 b1)  [0.001]
0.014: (move-robot r2 lp l4)  [0.001]
0.015: (serve-cup r2 p4 l4 t4 b1)  [0.001]
0.016: (move-robot r2 l4 l3)  [0.001]
0.017: (get-cup r2 a1 l3 t3 b1)  [0.001]
0.018: (move-robot r2 l3 lp)  [0.001]
0.019: (give-cup r2 r1 t3 lp b1 b2)  [0.001]
0.020: (move-robot r1 lp l1)  [0.001]
0.021: (fill-cup r1 m1 l1 t3 b2)  [0.001]
0.022: (move-robot r1 l1 lp)  [0.001]
0.023: (give-cup r1 r2 t3 lp b2 b1)  [0.001]
0.024: (move-robot r2 lp l4)  [0.001]
0.025: (serve-cup r2 p3 l4 t3 b1)  [0.001]
0.026: (move-robot r2 l4 lp)  [0.001]
0.027: (give-cup r2 r1 t5 lp b2 b1)  [0.001]
0.028: (move-robot r1 lp l1)  [0.001]
0.029: (fill-cup r1 m1 l1 t5 b1)  [0.001]
0.030: (move-robot r1 l1 lp)  [0.001]
0.031: (give-cup r1 r2 t5 lp b1 b2)  [0.001]
0.032: (move-robot r2 lp l3)  [0.001]
0.033: (serve-cup r2 p5 l3 t5 b2)  [0.001]
```

Estos resultados son coherentes con los de la primera instancia, ya que el orden de planificadores, de mejor a peor, sigue siendo Optic, LPG y FF. Además, tiene sentido que los planes tengan más pasos, puesto que hay más usuarios a los que servir tazas de té, de forma que habrá que realizar más acciones como coger y llenar tazas.

La segunda modificación consistía en quitar un robot. Por tanto, nos hemos quedado simplemente con el robot r1, de forma que debemos explicitar que sea capaz de acceder a todas las habitaciones, si no, el robot no podrá acceder a la habitación donde está el armario de tazas y los planificadores no obtendrán una solución. Los resultados son planes que alcanzan el objetivo del problema y han sido:

- Con FF, un plan de 36 pasos:

```
    ff: found legal plan as follows
step    0: MOVE-ROBOT R1 L1 LP
        1: MOVE-ROBOT R1 LP L3
        2: GET-CUP R1 A1 L3 T1 B1
        3: GIVE-CUP R1 R1 T1 L3 B1 B2
        4: GET-CUP R1 A1 L3 T2 B1
        5: MOVE-ROBOT R1 L3 LP
        6: MOVE-ROBOT R1 LP L1
        7: FILL-CUP R1 M1 L1 T1 B2
        8: FILL-CUP R1 M1 L1 T2 B1
        9: MOVE-ROBOT R1 L1 L2
       10: SERVE-CUP R1 P1 L2 T1 B2
       11: MOVE-ROBOT R1 L2 LP
       12: MOVE-ROBOT R1 LP L1
```

```
        13: SERVE-CUP R1 P2 L1 T2 B1
        14: MOVE-ROBOT R1 L1 LP
        15: MOVE-ROBOT R1 LP L3
        16: GET-CUP R1 A1 L3 T3 B1
        17: GIVE-CUP R1 R1 T3 L3 B1 B2
        18: GET-CUP R1 A1 L3 T4 B1
        19: MOVE-ROBOT R1 L3 LP
        20: MOVE-ROBOT R1 LP L1
        21: FILL-CUP R1 M1 L1 T4 B1
        22: FILL-CUP R1 M1 L1 T3 B2
        23: MOVE-ROBOT R1 L1 LP
        24: MOVE-ROBOT R1 LP L4
        25: SERVE-CUP R1 P3 L4 T3 B2
        26: SERVE-CUP R1 P4 L4 T4 B1
        27: MOVE-ROBOT R1 L4 LP
        28: MOVE-ROBOT R1 LP L3
        29: GET-CUP R1 A1 L3 T5 B1
        30: MOVE-ROBOT R1 L3 LP
        31: MOVE-ROBOT R1 LP L1
        32: FILL-CUP R1 M1 L1 T5 B1
        33: MOVE-ROBOT R1 L1 LP
        34: MOVE-ROBOT R1 LP L3
        35: SERVE-CUP R1 P5 L3 T5 B1
```

- Con LPG, un plan de 36 pasos:

```
    ; Time 0.12
; Search time 0.11
; Parsing time 0.00
; Mutex time 0.01
; NrActions 38

0:   (MOVE-ROBOT R1 L1 LP) [1]
1:   (MOVE-ROBOT R1 LP L3) [1]
2:   (GET-CUP R1 A1 L3 T1 B2) [1]
3:   (MOVE-ROBOT R1 L3 LP) [1]
4:   (MOVE-ROBOT R1 LP L1) [1]
5:   (FILL-CUP R1 M1 L1 T1 B2) [1]
6:   (MOVE-ROBOT R1 L1 L2) [1]
7:   (SERVE-CUP R1 P1 L2 T1 B2) [1]
8:   (MOVE-ROBOT R1 L2 LP) [1]
9:   (MOVE-ROBOT R1 LP L3) [1]
10:  (GET-CUP R1 A1 L3 T3 B2) [1]
10:  (GET-CUP R1 A1 L3 T5 B1) [1]
11:  (MOVE-ROBOT R1 L3 L4) [1]
12:  (MOVE-ROBOT R1 L4 LP) [1]
13:  (MOVE-ROBOT R1 LP L1) [1]
14:  (FILL-CUP R1 M1 L1 T5 B1) [1]
15:  (MOVE-ROBOT R1 L1 LP) [1]
16:  (MOVE-ROBOT R1 LP L3) [1]
17:  (SERVE-CUP R1 P5 L3 T5 B1) [1]
18:  (GET-CUP R1 A1 L3 T2 B1) [1]
19:  (MOVE-ROBOT R1 L3 LP) [1]
20:  (MOVE-ROBOT R1 LP L1) [1]
21:  (FILL-CUP R1 M1 L1 T3 B2) [1]
22:  (MOVE-ROBOT R1 L1 LP) [1]
23:  (MOVE-ROBOT R1 LP L4) [1]
24:  (SERVE-CUP R1 P3 L4 T3 B2) [1]
25:  (MOVE-ROBOT R1 L4 L3) [1]
26:  (GET-CUP R1 A1 L3 T4 B2) [1]
27:  (MOVE-ROBOT R1 L3 LP) [1]
28:  (MOVE-ROBOT R1 LP L1) [1]
29:  (FILL-CUP R1 M1 L1 T2 B1) [1]
29:  (FILL-CUP R1 M1 L1 T4 B2) [1]
```

```
30:   (SERVE-CUP R1 P2 L1 T2 B1) [1]
31:   (MOVE-ROBOT R1 L1 LP) [1]
32:   (MOVE-ROBOT R1 LP L3) [1]
33:   (MOVE-ROBOT R1 L3 LP) [1]
34:   (MOVE-ROBOT R1 LP L4) [1]
35:   (SERVE-CUP R1 P4 L4 T4 B2) [1]
```

- Con Optic, un plan de 29 pasos:

```
    ; Plan found with metric 0.028
; States evaluated so far: 91
; States pruned based on pre-heuristic cost lower bound: 0
; Time 0.13
0.000: (move-robot r1 l1 lp)  [0.001]
0.001: (move-robot r1 lp l3)  [0.001]
0.002: (get-cup r1 a1 l3 t5 b1)  [0.001]
0.002: (get-cup r1 a1 l3 t4 b2)  [0.001]
0.003: (move-robot r1 l3 lp)  [0.001]
0.004: (move-robot r1 lp l1)  [0.001]
0.005: (fill-cup r1 m1 l1 t5 b1)  [0.001]
0.005: (fill-cup r1 m1 l1 t4 b2)  [0.001]
0.006: (move-robot r1 l1 lp)  [0.001]
0.007: (move-robot r1 lp l3)  [0.001]
0.008: (serve-cup r1 p5 l3 t5 b1)  [0.001]
0.009: (move-robot r1 l3 l4)  [0.001]
0.010: (serve-cup r1 p4 l4 t4 b2)  [0.001]
0.011: (move-robot r1 l4 l3)  [0.001]
0.012: (get-cup r1 a1 l3 t3 b1)  [0.001]
0.012: (get-cup r1 a1 l3 t2 b2)  [0.001]
0.013: (move-robot r1 l3 lp)  [0.001]
0.014: (move-robot r1 lp l1)  [0.001]
0.015: (fill-cup r1 m1 l1 t3 b1)  [0.001]
0.015: (fill-cup r1 m1 l1 t2 b2)  [0.001]
0.016: (serve-cup r1 p2 l1 t2 b2)  [0.001]
0.017: (move-robot r1 l1 lp)  [0.001]
0.018: (move-robot r1 lp l4)  [0.001]
0.019: (serve-cup r1 p3 l4 t3 b1)  [0.001]
0.020: (move-robot r1 l4 lp)  [0.001]
0.021: (move-robot r1 lp l3)  [0.001]
0.022: (get-cup r1 a1 l3 t1 b1)  [0.001]
0.023: (move-robot r1 l3 lp)  [0.001]
0.024: (move-robot r1 lp l2)  [0.001]
0.025: (move-robot r1 l2 l1)  [0.001]
0.026: (fill-cup r1 m1 l1 t1 b1)  [0.001]
0.027: (move-robot r1 l1 l2)  [0.001]
0.028: (serve-cup r1 p1 l2 t1 b1)  [0.001]

 * All goal deadlines now no later than 0.028
```

Apreciando estos resultados, podemos observar que el hecho de quitar robots no afecta significativamente a que el plan obtenido sea más eficiente. De hecho, en el caso de Optic, vemos que con un solo robot el plan se ejecuta en menos pasos que teniendo dos robots.

La última modificación ha consistido en añadir un tercer robot a los dos que ya teníamos. Este tercer robot (al que denominamos r3) tiene permitido acceder a la habitación 3, la 4 y el pasillo, igual que el robot r2. Los resultados han sido planes que alcanzan el objetivo del problema y son:

- Con FF, un plan de 49 pasos:

```
    ff: found legal plan as follows
step    0: GET-CUP R2 A1 L3 T2 B1
        1: MOVE-ROBOT R2 L3 LP
        2: MOVE-ROBOT R1 L1 LP
        3: GIVE-CUP R2 R1 T2 LP B1 B2
        4: MOVE-ROBOT R1 LP L1
        5: FILL-CUP R1 M1 L1 T2 B2
        6: GIVE-CUP R1 R1 T2 L1 B2 B1
        7: SERVE-CUP R1 P2 L1 T2 B1
        8: MOVE-ROBOT R1 L1 LP
        9: MOVE-ROBOT R2 LP L3
       10: GET-CUP R2 A1 L3 T1 B1
       11: MOVE-ROBOT R2 L3 LP
       12: GIVE-CUP R2 R1 T1 LP B1 B2
       13: MOVE-ROBOT R1 LP L1
       14: FILL-CUP R1 M1 L1 T1 B2
       15: MOVE-ROBOT R1 L1 L2
       16: SERVE-CUP R1 P1 L2 T1 B2
       17: MOVE-ROBOT R1 L2 LP
       18: MOVE-ROBOT R2 LP L3
       19: GET-CUP R2 A1 L3 T3 B1
       20: MOVE-ROBOT R2 L3 LP
       21: GIVE-CUP R2 R1 T3 LP B1 B2
       22: MOVE-ROBOT R1 LP L1
       23: FILL-CUP R1 M1 L1 T3 B2
       24: MOVE-ROBOT R1 L1 LP
       25: GIVE-CUP R1 R2 T3 LP B2 B1
       26: MOVE-ROBOT R2 LP L4
       27: SERVE-CUP R2 P3 L4 T3 B1
       28: MOVE-ROBOT R2 L4 L3
       29: GET-CUP R2 A1 L3 T4 B1
       30: MOVE-ROBOT R2 L3 LP
       31: GIVE-CUP R2 R1 T4 LP B1 B2
       32: MOVE-ROBOT R1 LP L1
       33: FILL-CUP R1 M1 L1 T4 B2
       34: MOVE-ROBOT R1 L1 LP
       35: GIVE-CUP R1 R2 T4 LP B2 B1
       36: MOVE-ROBOT R2 LP L4
       37: SERVE-CUP R2 P4 L4 T4 B1
       38: MOVE-ROBOT R2 L4 LP
       39: MOVE-ROBOT R2 LP L3
       40: GET-CUP R2 A1 L3 T5 B1
       41: MOVE-ROBOT R2 L3 LP
       42: GIVE-CUP R2 R1 T5 LP B1 B1
       43: MOVE-ROBOT R1 LP L1
       44: FILL-CUP R1 M1 L1 T5 B1
       45: MOVE-ROBOT R1 L1 LP
       46: GIVE-CUP R1 R2 T5 LP B1 B1
       47: MOVE-ROBOT R2 LP L3
       48: SERVE-CUP R2 P5 L3 T5 B1
```

- Con LPG, un plan de 69 pasos:

```
    ; Time 1.08
; Search time 1.05
; Parsing time 0.03
; Mutex time 0.00
; NrActions 92

0:    (MOVE-ROBOT R1 L1 LP) [1]
0:    (MOVE-ROBOT R3 L4 L3) [1]
0:    (GET-CUP R2 A1 L3 T2 B2) [1]
0:    (GET-CUP R2 A1 L3 T3 B1) [1]
```

```
1:   (GET-CUP R3 A1 L3 T1 B1) [1]
1:   (MOVE-ROBOT R2 L3 L4) [1]
2:   (MOVE-ROBOT R3 L3 L4) [1]
2:   (MOVE-ROBOT R2 L4 LP) [1]
3:   (MOVE-ROBOT R3 L4 LP) [1]
4:   (MOVE-ROBOT R3 LP L3) [1]
5:   (MOVE-ROBOT R3 L3 LP) [1]
6:   (GIVE-CUP R3 R1 T1 LP B1 B2) [1]
7:   (MOVE-ROBOT R1 LP L1) [1]
7:   (GIVE-CUP R2 R3 T2 LP B2 B1) [1]
8:   (FILL-CUP R1 M1 L1 T1 B2) [1]
9:   (MOVE-ROBOT R1 L1 L2) [1]
10:  (SERVE-CUP R1 P1 L2 T1 B2) [1]
11:  (MOVE-ROBOT R1 L2 LP) [1]
12:  (GIVE-CUP R3 R1 T2 LP B1 B2) [1]
13:  (MOVE-ROBOT R1 LP L1) [1]
14:  (FILL-CUP R1 M1 L1 T2 B2) [1]
15:  (SERVE-CUP R1 P2 L1 T2 B2) [1]
16:  (MOVE-ROBOT R1 L1 LP) [1]
17:  (GIVE-CUP R2 R1 T3 LP B1 B2) [1]
18:  (MOVE-ROBOT R1 LP L1) [1]
19:  (FILL-CUP R1 M1 L1 T3 B2) [1]
20:  (MOVE-ROBOT R1 L1 L2) [1]
21:  (MOVE-ROBOT R1 L2 LP) [1]
22:  (GIVE-CUP R1 R2 T3 LP B2 B1) [1]
23:  (GIVE-CUP R2 R3 T3 LP B1 B2) [1]
24:  (GIVE-CUP R3 R2 T3 LP B2 B1) [1]
25:  (GIVE-CUP R2 R3 T3 LP B1 B2) [1]
26:  (GIVE-CUP R3 R1 T3 LP B2 B1) [1]
27:  (GIVE-CUP R1 R3 T3 LP B1 B2) [1]
28:  (GIVE-CUP R3 R2 T3 LP B2 B1) [1]
29:  (GIVE-CUP R2 R3 T3 LP B1 B2) [1]
30:  (MOVE-ROBOT R3 LP L3) [1]
30:  (MOVE-ROBOT R2 LP L3) [1]
31:  (GIVE-CUP R3 R2 T3 L3 B2 B1) [1]
32:  (MOVE-ROBOT R3 L3 LP) [1]
32:  (MOVE-ROBOT R2 L3 LP) [1]
33:  (GIVE-CUP R2 R3 T3 LP B1 B2) [1]
34:  (GIVE-CUP R3 R2 T3 LP B2 B1) [1]
35:  (MOVE-ROBOT R2 LP L4) [1]
35:  (MOVE-ROBOT R3 LP L3) [1]
36:  (MOVE-ROBOT R2 L4 L3) [1]
37:  (GIVE-CUP R2 R3 T3 L3 B1 B2) [1]
38:  (GIVE-CUP R3 R2 T3 L3 B2 B1) [1]
39:  (MOVE-ROBOT R3 L3 LP) [1]
39:  (MOVE-ROBOT R2 L3 LP) [1]
40:  (GIVE-CUP R2 R3 T3 LP B1 B2) [1]
41:  (GIVE-CUP R3 R1 T3 LP B2 B1) [1]
41:  (MOVE-ROBOT R2 LP L4) [1]
42:  (GIVE-CUP R1 R3 T3 LP B1 B2) [1]
43:  (MOVE-ROBOT R3 LP L4) [1]
44:  (GIVE-CUP R3 R2 T3 L4 B2 B1) [1]
45:  (MOVE-ROBOT R3 L4 L3) [1]
45:  (MOVE-ROBOT R2 L4 L3) [1]
46:  (GIVE-CUP R2 R3 T3 L3 B1 B2) [1]
47:  (MOVE-ROBOT R3 L3 LP) [1]
47:  (MOVE-ROBOT R2 L3 LP) [1]
48:  (GIVE-CUP R3 R2 T3 LP B2 B1) [1]
49:  (GIVE-CUP R2 R3 T3 LP B1 B2) [1]
50:  (GIVE-CUP R3 R1 T3 LP B2 B1) [1]
50:  (MOVE-ROBOT R2 LP L3) [1]
51:  (GIVE-CUP R1 R3 T3 LP B1 B2) [1]
51:  (GET-CUP R2 A1 L3 T5 B1) [1]
52:  (MOVE-ROBOT R3 LP L4) [1]
```

```
52:   (MOVE-ROBOT R2 L3 L4) [1]
53:   (SERVE-CUP R3 P3 L4 T3 B2) [1]
54:   (GIVE-CUP R2 R3 T5 L4 B1 B2) [1]
55:   (MOVE-ROBOT R2 L4 L3) [1]
55:   (MOVE-ROBOT R3 L4 LP) [1]
56:   (GET-CUP R2 A1 L3 T4 B1) [1]
56:   (GIVE-CUP R3 R1 T5 LP B2 B1) [1]
57:   (MOVE-ROBOT R2 L3 LP) [1]
58:   (GIVE-CUP R2 R1 T4 LP B1 B2) [1]
59:   (GIVE-CUP R1 R3 T4 LP B2 B1) [1]
60:   (GIVE-CUP R3 R1 T4 LP B1 B2) [1]
61:   (MOVE-ROBOT R1 LP L1) [1]
62:   (FILL-CUP R1 M1 L1 T4 B2) [1]
62:   (FILL-CUP R1 M1 L1 T5 B1) [1]
63:   (MOVE-ROBOT R1 L1 LP) [1]
64:   (GIVE-CUP R1 R2 T5 LP B1 B2) [1]
64:   (GIVE-CUP R1 R3 T4 LP B2 B1) [1]
65:   (MOVE-ROBOT R2 LP L3) [1]
65:   (MOVE-ROBOT R3 LP L3) [1]
66:   (MOVE-ROBOT R2 L3 L4) [1]
66:   (MOVE-ROBOT R3 L3 L4) [1]
67:   (MOVE-ROBOT R2 L4 L3) [1]
67:   (SERVE-CUP R3 P4 L4 T4 B1) [1]
68:   (SERVE-CUP R2 P5 L3 T5 B2) [1]
```

- Con Optic, un plan de 25 pasos:

```
    ; Plan found with metric 0.025
; States evaluated so far: 1049
; States pruned based on pre-heuristic cost lower bound: 0
; Time 1.42
0.000: (get-cup r2 a1 l3 t2 b2)  [0.001]
0.000: (move-robot r3 l4 lp)  [0.001]
0.000: (get-cup r2 a1 l3 t4 b1)  [0.001]
0.000: (move-robot r1 l1 lp)  [0.001]
0.001: (move-robot r2 l3 l4)  [0.001]
0.001: (move-robot r3 lp l3)  [0.001]
0.002: (get-cup r3 a1 l3 t5 b2)  [0.001]
0.002: (get-cup r3 a1 l3 t3 b1)  [0.001]
0.002: (move-robot r2 l4 lp)  [0.001]
0.003: (give-cup r2 r1 t2 lp b2 b1)  [0.001]
0.003: (move-robot r3 l3 lp)  [0.001]
0.004: (move-robot r2 lp l3)  [0.001]
0.004: (move-robot r1 lp l1)  [0.001]
0.005: (get-cup r2 a1 l3 t1 b2)  [0.001]
0.005: (fill-cup r1 m1 l1 t2 b1)  [0.001]
0.006: (move-robot r2 l3 lp)  [0.001]
0.006: (serve-cup r1 p2 l1 t2 b1)  [0.001]
0.007: (move-robot r1 l1 lp)  [0.001]
0.008: (give-cup r2 r1 t4 lp b1 b2)  [0.001]
0.008: (give-cup r3 r1 t5 lp b2 b1)  [0.001]
0.009: (move-robot r1 lp l1)  [0.001]
0.010: (fill-cup r1 m1 l1 t5 b1)  [0.001]
0.011: (move-robot r1 l1 lp)  [0.001]
0.012: (give-cup r1 r3 t5 lp b1 b2)  [0.001]
0.012: (give-cup r1 r2 t4 lp b2 b1)  [0.001]
0.013: (give-cup r2 r1 t1 lp b2 b1)  [0.001]
0.013: (move-robot r3 lp l3)  [0.001]
0.014: (serve-cup r3 p5 l3 t5 b2)  [0.001]
0.014: (move-robot r1 lp l1)  [0.001]
0.015: (move-robot r3 l3 lp)  [0.001]
0.015: (fill-cup r1 m1 l1 t1 b1)  [0.001]
0.016: (move-robot r1 l1 l2)  [0.001]
0.016: (give-cup r2 r3 t4 lp b1 b2)  [0.001]
```

```
38 0.017: (serve-cup r1 p1 l2 t1 b1) [0.001]
39 0.017: (move-robot r2 lp l4) [0.001]
40 0.018: (move-robot r1 l2 lp) [0.001]
41 0.018: (move-robot r2 l4 lp) [0.001]
42 0.019: (give-cup r3 r1 t3 lp b1 b2) [0.001]
43 0.019: (give-cup r3 r1 t4 lp b2 b1) [0.001]
44 0.020: (move-robot r1 lp l1) [0.001]
45 0.021: (fill-cup r1 m1 l1 t4 b1) [0.001]
46 0.021: (fill-cup r1 m1 l1 t3 b2) [0.001]
47 0.022: (move-robot r1 l1 lp) [0.001]
48 0.023: (give-cup r1 r2 t4 lp b1 b2) [0.001]
49 0.023: (give-cup r1 r2 t3 lp b2 b1) [0.001]
50 0.024: (move-robot r2 lp l4) [0.001]
51 0.025: (serve-cup r2 p4 l4 t4 b2) [0.001]
52 0.025: (serve-cup r2 p3 l4 t3 b1) [0.001]
```

A raíz de estos resultados, comprobamos que el hecho de añadir o quitar robots no afecta a que los planes obtenidos sean más eficientes, ya que hemos visto casos de planificadores que empeoran o mejoran el resultado sin concordancia con los otros planificadores para la misma instancia.

En el caso de Optic, podemos apreciar que un dato que sí hace mejorar los resultados es que los robots tengan menos restricciones a la hora de acceder a otras habitaciones. En la segunda instancia alternativa, el robot r1 no tenía ninguna de estas restricciones y el plan obtenido ha sido el mejor de los que hemos visto en este apartado. Mientras tanto, cuando teníamos tres robots operativos, el segundo y el tercero tenían las mismas restricciones de no acceder a las habitaciones 1 y 2, por lo que el plan obtenido era mejor que con dos robots, pero inferior al de un solo robot que puede moverse libremente entre habitaciones.

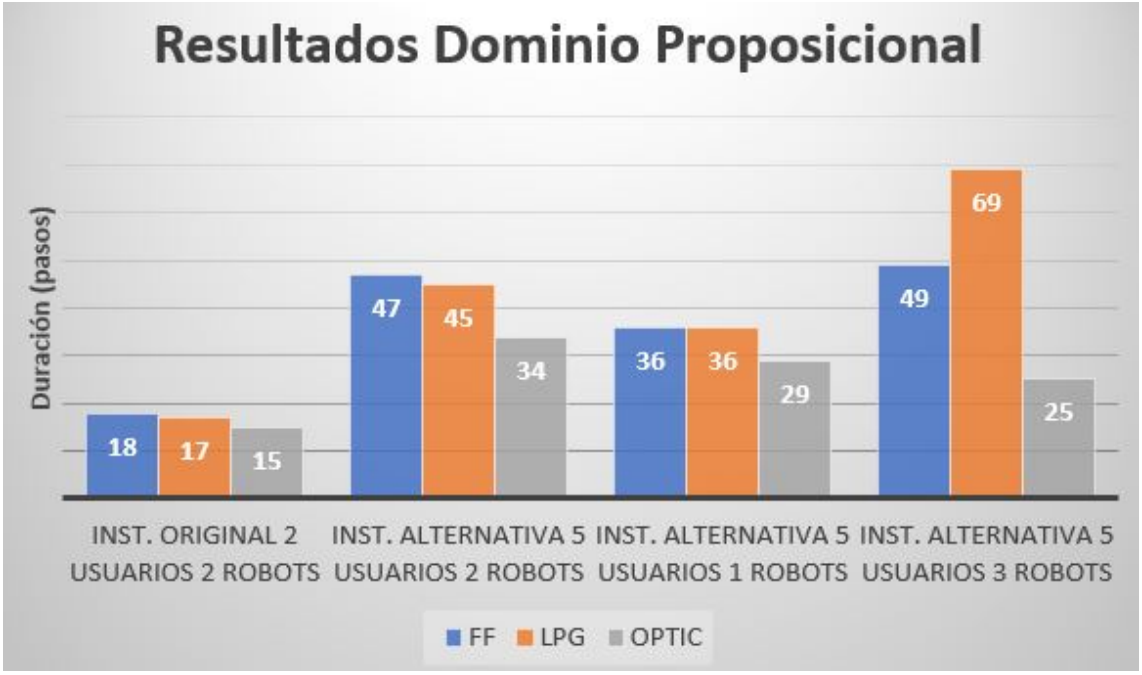

Figura 2.1: Gráfica comparativa con los resultados de los tres planificadores.

CAPÍTULO 3
Dominio Temporal

Partiendo del archivo *dominio_robots.pddl*, los cambios aplicados en el dominio proposicional para transitar a un dominio temporal han sido:

- Establecer la velocidad de los robots, para que un robot sea más rápido que otro. La velocidad solo afectará a los desplazamientos del robot.
- Establecer las distancias entre las habitaciones y las habitaciones y el pasillo, además de definir la duración de las conexiones en función de la velocidad de los robots y las distancias entre conexiones.
- Definir los pesos de las tazas en función de si están llenas o vacías. Esto se aplica en la acción de pasar una taza a un robot a una persona, para que la duración sea mayor en caso de que la taza esté llena.
- Establecer una duración fija a las acciones de hacer té y coger taza: 2 unidades de tiempo cada una.

Esto se ha incluido en el dominio mediante el uso de *:functions* que establecen variables que se irán modificando a lo largo de la ejecución del plan. Estas han sido:

```
(:functions
    (velocidad ?r - robot)
    (distancia ?o - lugar ?d - lugar)
    (peso ?t - taza)
)
```

Para definir el dominio temporal, hemos tenido que aplicar algunos cambios con respecto a la codificación STRIPS del dominio proposicional. El más significativo ha sido cambiar las *:actions* por *:durative-actions*. Esto implica que todas nuestras acciones ahora tienen una determinada duración.

3.1 Acciones en el dominio temporal

La acción de mover un robot ahora tiene la siguiente forma:

```
(:durative-action move-robot
       :parameters (?r - robot  ?o - lugar ?d - lugar)
       :duration (= ?duration (/ (distancia ?o ?d) (velocidad ?r)))
```

```
    :condition (and (at start (at ?r ?o)) (over all (linked ?o ?d)) (over all
        (allowed ?r ?d)))
        :effect
              (and
           (at end (not (at ?r ?o)))
         (at end (at ?r ?d))
        )
  )
```

Donde la duración de la acción depende de forma proporcional a la distancia de la habitación de origen y destino y de manera inversamente proporcional de la velocidad del robot. En las condiciones, se han establecido los *over all* para indicar que esos predicados no se verán modificados durante la ejecución de la acción y con *at start* los que tienen que cumplirse al principio de la acción. Por último, en los efectos se han añadido los *at end* para indicar que, al final de la acción, se cumplirán esos predicados.

La acción de dar una taza, donde la duración viene determinada por el peso de la taza, ahora tiene la forma:

```
  (:durative-action give-cup
       :parameters (?r1 - robot ?r2 - robot ?t - taza ?l - lugar ?b1 - brazo ?
           b2 - brazo)
       :duration (= ?duration (peso ?t))
    :condition (and (at start (at ?r1 ?l)) (over all (at ?r1 ?l)) (at start (
        at ?r2 ?l)) (over all(at ?r2 ?l)) (at start (have ?t ?r1 ?b1)) (over
        all (have ?t ?r1 ?b1)) (at start (libre ?b2 ?r2)) (over all (libre ?b2
         ?r2)) (at start (ocupado ?b1 ?r1)) (over all (ocupado ?b1 ?r1)))
       :effect
          (and
          (at end (have ?t ?r2 ?b2))
          (at end (not (have ?t ?r1 ?b1)))
          (at end (not (ocupado ?b1 ?r1)))
          (at end (ocupado ?b2 ?r2))
          (at end (not (libre ?b2 ?r2)))
          (at end (libre ?b1 ?r1))
          (at end (at ?r1 ?l))
          (at end (at ?r2 ?l))
        )
  )
```

La acción de llenar taza, donde la duración es fija y de 2 unidades de tiempo, ahora tiene la forma:

```
  (:durative-action fill-cup
       :parameters (?r - robot ?m - maquina ?l - lugar ?t - taza ?b - brazo)
       :duration (= ?duration 2)
    :condition (and (at start (at ?r ?l)) (over all (at ?r ?l))(over all (at ?
        m ?l))(at start (have ?t ?r ?b))(over all (have ?t ?r ?b))(at start (
        empty ?t))(over all (empty ?t)))
       :effect
          (and
          (at end (not (empty ?t)))
          (at end (full ?t))
          (at end (increase (peso ?t) (+ (peso ?t) 1)))
          (at end (at ?r ?l))
          (at end (have ?t ?r ?b))
        )
  )
```

Al final de la acción, es necesario incrementar el peso de la taza debido a que ahora está llena. En nuestro caso, hemos establecido que las tazas llenas tienen un peso de una 1 unidad por encima de las tazas vacías. Fíjese además que no es necesario añadir un (at start (at ?m ?l)) en las condiciones porque no hay ninguna acción para mover la máquina de té ni el armario.

La acción de servir una taza, igual que la acción de dar una taza, ahora tiene una duración determinada por el peso de esta:

```
(:durative-action serve-cup
      :parameters (?r - robot ?p - persona ?l - lugar ?t - taza ?b - brazo)
      :duration (= ?duration (peso ?t))
    :condition (and (at start (at ?r ?l))(over all (at ?r ?l)) (at start (at ?
        p ?l)) (over all (at ?p ?l)) (at start (have ?t ?r ?b)) (over all (
        have ?t ?r ?b)) (at start (full ?t)) (over all (full ?t)) (at start (
        ocupado ?b ?r)) (over all (ocupado ?b ?r)))
      :effect
         (and
         (at end (not (have ?t ?r ?b)))
         (at end (on ?t ?p))
         (at end (not (ocupado ?b ?r)))
         (at end (libre ?b ?r))
         (at end (at ?r ?l))
         (at end (at ?p ?l))
       )
)
```

Por último, la acción de coger una taza, igual que la acción de llenar la taza, tiene una duración fija de 2 unidades de tiempo:

```
(:durative-action get-cup
      :parameters (?r - robot ?a - armario ?l - lugar ?t - taza ?b - brazo)
      :duration (= ?duration 2)
    :condition (and (at start (at ?r ?l)) (over all (at ?r ?l)) (over all (at
        ?a ?l)) (at start (on ?t ?a)) (over all (on ?t ?a)) (at start (libre ?
        b ?r)) (over all (libre ?b ?r)))
      :effect
         (and
         (at end (not (on ?t ?a)))
         (at end (have ?t ?r ?b))
         (at end (ocupado ?b ?r))
         (at end (not (libre ?b ?r)))
         (at end (at ?r ?l))
       )
)
```

3.2 Nueva instancia del problema

Se ha redefinido la instancia original del problema para incluir las modificaciones del dominio temporal, partiendo del fichero *problema_robots.pddl*.

Para establecer que un robot es más rápido es otro, hemos definido que r1 sea más lento que r2 añadiendo en *:init* las siguientes líneas de código:

```
(= (velocidad r1) 1)
(= (velocidad r2) 2)
```

Para establecer las distancias entre las habitaciones, hemos definido que las habitaciones contiguas tengan una distancia menor entre ellas que con el pasillo añadiendo las siguientes líneas de código:

```
(= (distancia l1 l2) 2)
(= (distancia l2 l1) 2)

(= (distancia l1 lp) 4)
(= (distancia lp l1) 4)

(= (distancia l2 lp) 4)
(= (distancia lp l2) 4)

(= (distancia l3 l4) 2)
(= (distancia l4 l3) 2)

(= (distancia l3 lp) 4)
(= (distancia lp l3) 4)

(= (distancia l4 lp) 4)
(= (distancia lp l4) 4)
```

De esta forma, se añaden distancias (y se han de duplicar para establecer la bidireccionalidad entre las habitaciones) y se omiten aquellas distancias entre habitaciones no contiguas (ya que no es posible acceder de una a otra).

Para establecer los pesos de las tazas, hemos definido que cuando están vacías tengan un peso de 1 unidad (así cuando estén llenas tendrán un peso de 2, como hemos visto en las acciones del dominio temporal) añadiendo las siguientes líneas de código:

```
(= (peso t1) 1)
(= (peso t2) 1)
```

Por último, especificamos que queremos minimizar el coste del plan en función de la duración de las acciones con la siguiente línea de código:

```
(:metric minimize (total-time))
```

3.3 Resultados obtenidos

Cuando ejecutamos el dominio con la instancia anteriormente explicada en LPG, obtenemos el siguiente plan:

```
; Time 0.03
; Search time 0.03
; Parsing time 0.00
; Mutex time 0.00
; MakeSpan 21.00


0.0002:    (GET-CUP R2 A1 L3 T1 B2) [2.0000]
0.0002:    (MOVE-ROBOT R1 L1 LP) [4.0000]
0.0002:    (GET-CUP R2 A1 L3 T2 B1) [2.0000]
2.0004:    (MOVE-ROBOT R2 L3 LP) [2.0000]
```

```
4.0006:   (GIVE-CUP R2 R1 T1 LP B2 B1) [1.0000]
4.0006:   (GIVE-CUP R2 R1 T2 LP B1 B2) [1.0000]
4.0004:   (MOVE-ROBOT R1 LP L1) [4.0000]
8.0006:   (FILL-CUP R1 M1 L1 T2 B2) [2.0000]
8.0006:   (MOVE-ROBOT R1 L1 LP) [4.0000]
12.0008:   (GIVE-CUP R1 R2 T2 LP B2 B1) [3.0000]
13.0012:   (MOVE-ROBOT R2 LP L3) [2.0000]
15.0014:   (SERVE-CUP R2 P2 L3 T2 B1) [3.0000]
12.0008:   (MOVE-ROBOT R1 LP L1) [4.0000]
16.0010:   (FILL-CUP R1 M1 L1 T1 B1) [2.0000]
16.0014:   (MOVE-ROBOT R1 L1 L2) [2.0000]
18.0016:   (SERVE-CUP R1 P1 L2 T1 B1) [3.0000]
```

Es decir, nuestro plan ha tardado 21 unidades temporales en ejecutarse, ya que, aunque la suma de la duración de las acciones supera ese tiempo, algunas las realiza concurrentemente, por lo que consigue un resultado más eficiente.

Cuando ejecutamos el planificador Optic para la misma instancia, obtenemos:

```
; Plan found with metric 25.011
; Theoretical reachable cost 25.011
; States evaluated so far: 138
; States pruned based on pre-heuristic cost lower bound: 0
; Time 0.31
0.000: (get-cup r2 a1 l3 t2 b2)  [2.000]
0.000: (move-robot r1 l1 lp)  [4.000]
2.001: (get-cup r2 a1 l3 t1 b1)  [2.000]
4.002: (move-robot r2 l3 lp)  [2.000]
6.003: (give-cup r2 r1 t1 lp b1 b2)  [1.000]
6.003: (give-cup r2 r1 t2 lp b2 b1)  [1.000]
7.004: (move-robot r1 lp l2)  [4.000]
11.005: (move-robot r1 l2 l1)  [2.000]
13.006: (fill-cup r1 m1 l1 t1 b2)  [2.000]
13.006: (fill-cup r1 m1 l1 t2 b1)  [2.000]
13.007: (move-robot r1 l1 l2)  [2.000]
15.008: (serve-cup r1 p1 l2 t1 b2)  [3.000]
15.008: (move-robot r1 l2 lp)  [4.000]
19.009: (give-cup r1 r2 t2 lp b1 b2)  [3.000]
20.010: (move-robot r2 lp l3)  [2.000]
22.011: (serve-cup r2 p2 l3 t2 b2)  [3.000]

 * All goal deadlines now no later than 25.011
```

Lo que nos da un plan de 25 unidades temporales, algo peor que el obtenido con LPG. Esto se debe a que, mientras en LPG el robot r1 llenaba las dos tazas y le daba una al robot r2 antes de servir, en Optic llena las dos tazas, sirve la suya al usuario de la habitación 2 y vuelve al pasillo a entregarle la otra taza al robot r2.

3.4 Instancias alternativas

Hemos probado a modificar la instancia del problema para comprobar como afecta a los planes obtenidos. Es necesario comentar que todas las modificaciones parten de la anterior y solo la primera parte de la original.

Las modificaciones realizadas han sido las mismas que en el dominio proposicional, a excepción de que ahora tenemos 3 usuarios en lugar de 5. Esto se debe a que hicimos pruebas en Optic para 5 usuarios y el planificador se quedaba en bucle ejecutando sin arrojar ninguna solución.

3.4.1. Resultados para 2 robots y 3 usuarios

Para esta modificación, en LPG, hemos obtenido un plan de 37 unidades temporales:

```
; Time 0.03
; Search time 0.03
; Parsing time 0.00
; Mutex time 0.00
; MakeSpan 37.00


0.0002:    (GET-CUP R2 A1 L3 T2 B2) [2.0000]
0.0002:    (GET-CUP R2 A1 L3 T1 B1) [2.0000]
2.0004:    (MOVE-ROBOT R2 L3 LP) [2.0000]
0.0002:    (MOVE-ROBOT R1 L1 LP) [4.0000]
4.0006:    (GIVE-CUP R2 R1 T1 LP B1 B2) [1.0000]
4.0004:    (MOVE-ROBOT R1 LP L1) [4.0000]
8.0006:    (FILL-CUP R1 M1 L1 T1 B2) [2.0000]
8.0006:    (MOVE-ROBOT R1 L1 LP) [4.0000]
4.0006:    (MOVE-ROBOT R2 LP L3) [2.0000]
6.0008:    (MOVE-ROBOT R2 L3 LP) [2.0000]
12.0008:    (GIVE-CUP R2 R1 T2 LP B2 B1) [1.0000]
12.0008:    (MOVE-ROBOT R1 LP L1) [4.0000]
16.0010:    (FILL-CUP R1 M1 L1 T2 B1) [2.0000]
18.0012:    (SERVE-CUP R1 P2 L1 T2 B1) [3.0000]
11.0012:    (MOVE-ROBOT R2 LP L3) [2.0000]
13.0014:    (GET-CUP R2 A1 L3 T3 B2) [2.0000]
17.0016:    (MOVE-ROBOT R1 L1 LP) [4.0000]
13.0018:    (MOVE-ROBOT R2 L3 LP) [2.0000]
21.0018:    (GIVE-CUP R2 R1 T3 LP B2 B1) [1.0000]
22.0020:    (GIVE-CUP R1 R2 T3 LP B1 B2) [1.0000]
23.0022:    (GIVE-CUP R2 R1 T3 LP B2 B1) [1.0000]
21.0018:    (MOVE-ROBOT R1 LP L1) [4.0000]
25.0020:    (FILL-CUP R1 M1 L1 T3 B1) [2.0000]
25.0024:    (MOVE-ROBOT R1 L1 L2) [2.0000]
27.0026:    (SERVE-CUP R1 P1 L2 T1 B2) [3.0000]
27.0026:    (MOVE-ROBOT R1 L2 LP) [4.0000]
31.0028:    (GIVE-CUP R1 R2 T3 LP B1 B2) [3.0000]
32.0032:    (MOVE-ROBOT R2 LP L4) [2.0000]
34.0034:    (SERVE-CUP R2 P3 L4 T3 B2) [3.0000]
```

Mientras que en Optic, hemos obtenido un plan de 54 unidades temporales:

```
; Plan found with metric 54.022
; Theoretical reachable cost 54.022
; States evaluated so far: 408
; States pruned based on pre-heuristic cost lower bound: 0
; Time 1.49
0.000: (get-cup r2 a1 l3 t3 b2)  [2.000]
0.000: (move-robot r1 l1 lp)  [4.000]
2.001: (get-cup r2 a1 l3 t2 b1)  [2.000]
4.002: (move-robot r2 l3 lp)  [2.000]
6.003: (move-robot r2 lp l3)  [2.000]
6.003: (give-cup r2 r1 t2 lp b1 b2)  [1.000]
7.004: (move-robot r1 lp l2)  [4.000]
8.004: (get-cup r2 a1 l3 t1 b1)  [2.000]
10.005: (move-robot r2 l3 lp)  [2.000]
11.005: (move-robot r1 l2 l1)  [2.000]
13.006: (fill-cup r1 m1 l1 t2 b2)  [2.000]
15.007: (serve-cup r1 p2 l1 t2 b2)  [3.000]
16.008: (move-robot r1 l1 l2)  [2.000]
```

```
18.009: (move-robot r1 l2 lp)  [4.000]
22.010: (give-cup r2 r1 t1 lp b1 b2)  [1.000]
23.011: (move-robot r1 lp l2)  [4.000]
27.012: (move-robot r1 l2 l1)  [2.000]
29.013: (fill-cup r1 m1 l1 t1 b2)  [2.000]
29.014: (move-robot r1 l1 l2)  [2.000]
31.015: (serve-cup r1 p1 l2 t1 b2)  [3.000]
34.016: (move-robot r1 l2 lp)  [4.000]
38.017: (give-cup r2 r1 t3 lp b2 b1)  [1.000]
38.017: (move-robot r1 lp l1)  [4.000]
42.018: (fill-cup r1 m1 l1 t3 b1)  [2.000]
44.019: (move-robot r1 l1 lp)  [4.000]
48.020: (give-cup r1 r2 t3 lp b1 b2)  [3.000]
49.021: (move-robot r2 lp l4)  [2.000]
51.022: (serve-cup r2 p3 l4 t3 b2)  [3.000]

 * All goal deadlines now no later than 54.022
```

3.4.2. Resultados para 1 robot y 3 usuarios

En esta modificación, hemos obtenido, para LPG, un plan de 55 unidades temporales:

```
; Time 0.04
; Search time 0.04
; Parsing time 0.00
; Mutex time 0.00
; MakeSpan 55.00


0.0002:    (MOVE-ROBOT R1 L1 LP) [4.0000]
4.0004:    (MOVE-ROBOT R1 LP L3) [4.0000]
8.0006:    (GET-CUP R1 A1 L3 T1 B1) [2.0000]
8.0006:    (MOVE-ROBOT R1 L3 LP) [4.0000]
12.0008:    (MOVE-ROBOT R1 LP L1) [4.0000]
16.0010:    (FILL-CUP R1 M1 L1 T1 B1) [2.0000]
16.0014:    (MOVE-ROBOT R1 L1 L2) [2.0000]
18.0016:    (SERVE-CUP R1 P1 L2 T1 B1) [3.0000]
18.0016:    (MOVE-ROBOT R1 L2 LP) [4.0000]
22.0018:    (MOVE-ROBOT R1 LP L3) [4.0000]
26.0020:    (GET-CUP R1 A1 L3 T2 B1) [2.0000]
26.0020:    (GET-CUP R1 A1 L3 T3 B2) [2.0000]
26.0020:    (MOVE-ROBOT R1 L3 LP) [4.0000]
30.0022:    (MOVE-ROBOT R1 LP L1) [4.0000]
34.0024:    (FILL-CUP R1 M1 L1 T3 B2) [2.0000]
34.0024:    (MOVE-ROBOT R1 L1 LP) [4.0000]
38.0026:    (MOVE-ROBOT R1 LP L4) [4.0000]
42.0028:    (SERVE-CUP R1 P3 L4 T3 B2) [3.0000]
42.0028:    (MOVE-ROBOT R1 L4 LP) [4.0000]
46.0030:    (MOVE-ROBOT R1 LP L1) [4.0000]
50.0032:    (FILL-CUP R1 M1 L1 T2 B1) [2.0000]
52.0034:    (SERVE-CUP R1 P2 L1 T2 B1) [3.0000]
```

Mientras que en Optic obtenemos uno de 53 unidades temporales:

```
; Plan found with metric 53.016
; Theoretical reachable cost 53.016
; States evaluated so far: 567
; States pruned based on pre-heuristic cost lower bound: 3
; Time 1.83
```

```
0.000: (move-robot r1 l1 lp)  [4.000]
4.001: (move-robot r1 lp l3)  [4.000]
8.002: (get-cup r1 a1 l3 t2 b1)  [2.000]
10.003: (get-cup r1 a1 l3 t3 b2)  [2.000]
12.004: (move-robot r1 l3 lp)  [4.000]
16.005: (move-robot r1 lp l1)  [4.000]
20.006: (fill-cup r1 m1 l1 t3 b2)  [2.000]
22.007: (fill-cup r1 m1 l1 t2 b1)  [2.000]
24.008: (move-robot r1 l1 lp)  [4.000]
24.008: (serve-cup r1 p2 l1 t2 b1)  [3.000]
28.009: (move-robot r1 lp l3)  [4.000]
32.010: (move-robot r1 l3 lp)  [4.000]
32.010: (get-cup r1 a1 l3 t1 b1)  [2.000]
36.011: (move-robot r1 lp l1)  [4.000]
40.012: (fill-cup r1 m1 l1 t1 b1)  [2.000]
40.013: (move-robot r1 l1 l2)  [2.000]
42.014: (move-robot r1 l2 lp)  [4.000]
42.014: (serve-cup r1 p1 l2 t1 b1)  [3.000]
46.015: (move-robot r1 lp l4)  [4.000]
50.016: (serve-cup r1 p3 l4 t3 b2)  [3.000]

 * All goal deadlines now no later than 53.016
```

3.4.3. Resultados para 3 robots y 3 usuarios

Por último, en esta modificación hemos añadido un tercer robot r3 que es más veloz que r2 (hemos establecido su variable de velocidad a 3) y con ella obtenemos, en LPG, un plan de 55 unidades temporales:

```
; Time 0.03
; Search time 0.03
; Parsing time 0.00
; Mutex time 0.00
; MakeSpan 55.00


0.0002:    (GET-CUP R3 A1 L3 T3 B2) [2.0000]
0.0002:    (GET-CUP R2 A1 L3 T1 B1) [2.0000]
0.0002:    (MOVE-ROBOT R1 L1 LP) [4.0000]
0.0006:    (MOVE-ROBOT R2 L3 LP) [2.0000]
4.0004:    (GIVE-CUP R2 R1 T1 LP B1 B2) [1.0000]
4.0004:    (MOVE-ROBOT R1 LP L1) [4.0000]
8.0006:    (FILL-CUP R1 M1 L1 T1 B2) [2.0000]
8.0006:    (MOVE-ROBOT R1 L1 LP) [4.0000]
3.0008:    (MOVE-ROBOT R2 LP L3) [2.0000]
5.0010:    (GET-CUP R2 A1 L3 T2 B1) [2.0000]
5.0014:    (MOVE-ROBOT R2 L3 LP) [2.0000]
0.6673:    (MOVE-ROBOT R3 L3 LP) [1.3333]
12.0008:    (GIVE-CUP R1 R3 T1 LP B2 B1) [3.0000]
13.6679:    (MOVE-ROBOT R3 LP L4) [1.3333]
7.0016:    (MOVE-ROBOT R2 LP L4) [2.0000]
15.0014:    (GIVE-CUP R3 R2 T1 L4 B1 B2) [3.0000]
16.0018:    (MOVE-ROBOT R2 L4 LP) [2.0000]
16.6685:    (MOVE-ROBOT R3 L4 LP) [1.3333]
18.0020:    (GIVE-CUP R2 R3 T1 LP B2 B1) [3.0000]
21.0022:    (GIVE-CUP R3 R1 T1 LP B1 B2) [3.0000]
20.0026:    (MOVE-ROBOT R1 LP L2) [4.0000]
24.0028:    (SERVE-CUP R1 P1 L2 T1 B2) [3.0000]
22.6693:    (MOVE-ROBOT R3 LP L3) [1.3333]
24.0028:    (MOVE-ROBOT R3 L3 LP) [1.3333]
```

```
24.0028:    (MOVE-ROBOT R1 L2 LP) [4.0000]
28.0030:    (GIVE-CUP R3 R1 T3 LP B2 B1) [1.0000]
28.0030:    (GIVE-CUP R2 R1 T2 LP B1 B2) [1.0000]
28.0030:    (MOVE-ROBOT R1 LP L1) [4.0000]
32.0032:    (FILL-CUP R1 M1 L1 T2 B2) [2.0000]
32.0032:    (MOVE-ROBOT R1 L1 LP) [4.0000]
36.0034:    (MOVE-ROBOT R1 LP L2) [4.0000]
40.0036:    (MOVE-ROBOT R1 L2 L1) [2.0000]
42.0038:    (FILL-CUP R1 M1 L1 T3 B1) [2.0000]
42.0038:    (SERVE-CUP R1 P2 L1 T2 B2) [3.0000]
43.0042:    (MOVE-ROBOT R1 L1 L2) [2.0000]
45.0044:    (MOVE-ROBOT R1 L2 LP) [4.0000]
49.0046:    (GIVE-CUP R1 R2 T3 LP B1 B2) [3.0000]
50.0050:    (MOVE-ROBOT R2 LP L4) [2.0000]
52.0052:    (SERVE-CUP R2 P3 L4 T3 B2) [3.0000]
```

Mientras que en Optic hemos obtenido uno de 67 unidades temporales:

```
; Plan found with metric 67.020
; Theoretical reachable cost 67.020
; States evaluated so far: 2988
; States pruned based on pre-heuristic cost lower bound: 0
; Time 11.31
0.000: (get-cup r2 a1 l3 t3 b1)  [2.000]
0.000: (get-cup r3 a1 l3 t2 b2)  [2.000]
0.000: (move-robot r1 l1 lp)  [4.000]
2.001: (get-cup r2 a1 l3 t1 b2)  [2.000]
2.001: (move-robot r3 l3 lp)  [1.333]
2.002: (move-robot r2 l3 lp)  [2.000]
4.001: (move-robot r1 lp l2)  [4.000]
4.003: (give-cup r2 r1 t1 lp b2 b1)  [1.000]
4.003: (move-robot r2 lp l4)  [2.000]
6.004: (move-robot r2 l4 lp)  [2.000]
8.002: (move-robot r1 l2 lp)  [4.000]
8.005: (move-robot r2 lp l4)  [2.000]
10.006: (move-robot r2 l4 lp)  [2.000]
12.003: (move-robot r1 lp l1)  [4.000]
16.004: (fill-cup r1 m1 l1 t1 b1)  [2.000]
16.005: (move-robot r1 l1 l2)  [2.000]
18.006: (serve-cup r1 p1 l2 t1 b1)  [3.000]
21.007: (move-robot r1 l2 lp)  [4.000]
25.008: (move-robot r1 lp l1)  [4.000]
29.009: (move-robot r1 l1 lp)  [4.000]
33.010: (move-robot r1 lp l1)  [4.000]
37.011: (move-robot r1 l1 lp)  [4.000]
41.012: (move-robot r1 lp l1)  [4.000]
41.012: (give-cup r3 r1 t2 lp b2 b1)  [1.000]
42.013: (give-cup r2 r3 t3 lp b1 b2)  [1.000]
45.013: (fill-cup r1 m1 l1 t2 b1)  [2.000]
47.014: (move-robot r1 l1 lp)  [4.000]
47.014: (serve-cup r1 p2 l1 t2 b1)  [3.000]
51.015: (move-robot r1 lp l1)  [4.000]
51.015: (give-cup r3 r1 t3 lp b2 b1)  [1.000]
55.016: (fill-cup r1 m1 l1 t3 b1)  [2.000]
57.017: (move-robot r1 l1 lp)  [4.000]
61.018: (give-cup r1 r2 t3 lp b1 b2)  [3.000]
62.019: (move-robot r2 lp l4)  [2.000]
64.020: (serve-cup r2 p3 l4 t3 b2)  [3.000]

 * All goal deadlines now no later than 67.020
```

3.5 Conclusiones

En vista de los resultados obtenidos, podemos afirmar que, para este dominio temporal, el planificador más recomendable es LPG, ya que consigue mejores planes que Optic en la mayoría de los casos (el único en el que falla con respecto al otro obtiene una solución similar en coste).

Además, las variables que se han añadido en este dominio también hacen aumentar el coste de las acciones y, por tanto, el número de pasos que el plan ha de dar para alcanzar el objetivo del problema.

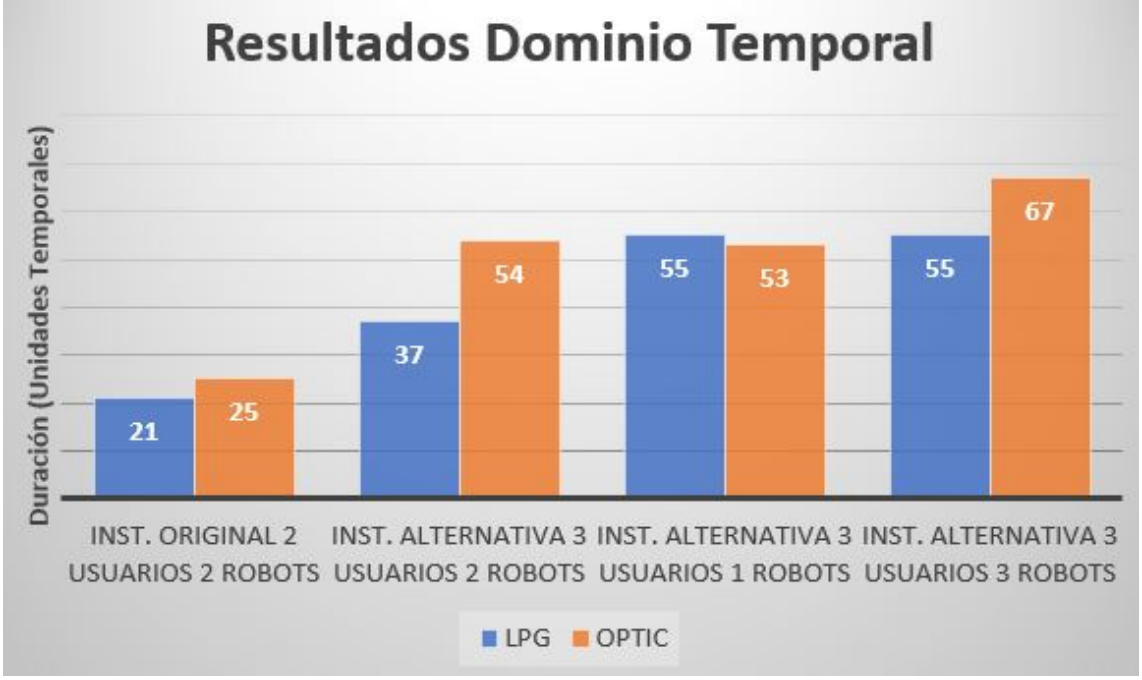

Figura 3.1: Gráfica comparativa con los resultados de LPG y Optic.

CAPÍTULO 4
Dominio con recursos numéricos

En este apartado, hemos definido una variable numérica que representa el consumo de un recurso, de forma que este consumo es inversamente proporcional al consumo de tiempo, es decir, a menos tiempo, más consumo del recurso.

En nuestro caso, el recurso es la electricidad de los robots. Para que solo haya un recurso, hemos definido un límite total de esta electricidad como un generador central del que se suministran todos los robots instanciados. Por tanto, a esta electricidad central se le resta la batería consumida por cada robot.

Para la realización de este dominio, hemos partido del fichero resultado de *dominio_robots.pddl* del dominio temporal.

4.1 Batería

Para introducir este componente en nuestro dominio, hemos añadido en *:functions (battery ?r - robot)* y *(total-battery-used)*. La primera indica la batería que tiene uno de nuestros robots, e irá decreciendo conforme se vayan sucediendo las acciones del plan. La segunda representa el total de batería que llevamos gastada en un determinado momento, de modo que cada vez que el robot gaste una cierta cantidad de batería, esa medida se sumará al total.

Lo que se intenta, pues, es minimizar, en la medida de lo posible, el incremento de la batería total consumida, indicándole además a las instancias del problema una cantidad de batería total que el planificador no puede sobrepasar.

Por otro lado, se han definido dos dominios. En uno de ellos, nuestro recurso recurso no es renovable, es decir, una vez se acaba la batería de los robots, estos no pueden realizar más acciones. En el otro, nuestro recurso es renovable, y hemos establecido un umbral mínimo de batería en el cuál cuando la batería del robot es inferior a este, el valor de la batería en ese robot pasa a ser un valor por encima del umbral, aunque no mucho mayor.

En los siguientes apartados explicamos cómo hemos implementado esto.

4.2 Nuevas acciones con el recurso numérico

Como hemos partido del dominio temporal para realizar el numérico, y dado que todas las acciones en el dominio son ejecutadas por los robots, el consumo del la batería viene determinado por el robot que realice la acción y la duración de dicha acción.

Así, en la acción *move-robot*, como esta tiene una duración de distancia entre la habitación origen y destino partido por la velocidad del robot que realiza la acción, esa función es la batería que gasta nuestro robot en esta acción.

Por tanto, hay que añadirle como condición a la acción que, al principio de esta, la batería del robot sea igual o mayor que esa función de distancia entre velocidad. Finalmente, en los efectos habrá que añadir que el robot ha perdido esa cantidad de batería y que el total de batería que llevamos gastada se incrementa en esa misma cantidad. De ahí que la acción pase a tener esta forma:

```
(:durative-action move-robot
       :parameters (?r - robot  ?o - lugar ?d - lugar)
       :duration (= ?duration (/ (distancia ?o ?d) (velocidad ?r)))
    :condition (and
           (at start (at ?r ?o))
           (over all (at ?r ?o))
           (over all (linked ?o ?d))
           (over all (allowed ?r ?d))
           (at start(>= (battery ?r)
             (/ 1 (/ (distancia ?o ?d) (velocidad ?r)))
           ))
         )
        :effect
             (and
           (at end (not (at ?r ?o)))
         (at end (at ?r ?d))
         (at end(increase (total-battery-used)
               (/ 1 (/ (distancia ?o ?d) (velocidad ?r)))
         ))
         (at end(decrease (battery ?r)
               (/ 1 (/ (distancia ?o ?d) (velocidad ?r)))
         ))
       )
  )
```

El mismo procedimiento se lleva a cabo para el resto de acciones del dominio.

Esto que acabamos de explicar es para el dominio con recursos numéricos no renovables. Para el dominio con recursos numéricos renovables, lo único que hemos añadido es una acción de más que se lleve a cabo cuando la batería del robot en cuestión es inferior a un umbral establecido; y que el efecto de la acción sea asignar a la batería del robot un valor ligeramente superior al umbral (hemos especificado 2 unidades por encima). Esta acción es:

```
(:durative-action renove
    :parameters (?r - robot)
    :duration (= ?duration 2)
    :condition (at start (< (battery ?r) (umbral)))
    :effect (at end (assign (battery ?r) (+ (umbral) 2)))
  )
```

4.3 Instancias optimizando el gasto de batería y el tiempo

Para las nuevas instancias del problema, simplemente hemos tenido que añadir al *:init* las siguientes líneas de código:

```
(= (total-battery-used) 0)
(= (battery r1) 150)
(= (battery r2) 150)
```

Y en el *:goal*:

```
(< (total-battery-used) 200)
```

De esta forma le asignamos al planificador un total de batería que no ha de superar y la batería inicial de nuestros robots (en el caso de 3 robots, simplemente es añadir un *battery* más para el tercero).

Finalmente, hemos de añadirle la siguiente métrica, para que minimice el total de batería gastada:

```
(:metric minimize (total-battery-used))
```

Para aquellas instancias en las que se nos pide minimizar el tiempo de las acciones, la métrica es:

```
(:metric minimize (total-time))
```

4.4 Comparación de resultados. Conclusiones

4.4.1. Plan para el recurso no renovable, minimizando la batería y con 2 robots

```
; Time 0.04
; Search time 0.04
; Parsing time 0.00
; Mutex time 0.00
; MetricValue 9.50

0.0002:   (GET-CUP R2 A1 L3 T1 B1) [2.0000]
0.0002:   (MOVE-ROBOT R1 L1 LP) [4.0000]
2.0006:   (GET-CUP R2 A1 L3 T2 B2) [2.0000]
4.0010:   (MOVE-ROBOT R2 L3 LP) [2.0000]
6.0012:   (GIVE-CUP R2 R1 T2 LP B2 B1) [1.0000]
7.0016:   (MOVE-ROBOT R2 LP L3) [2.0000]
9.0018:   (MOVE-ROBOT R2 L3 L4) [1.0000]
10.0020:   (MOVE-ROBOT R2 L4 LP) [2.0000]
12.0022:   (GIVE-CUP R2 R1 T1 LP B1 B2) [1.0000]
9.0026:   (MOVE-ROBOT R1 LP L1) [4.0000]
13.0028:   (FILL-CUP R1 M1 L1 T2 B1) [2.0000]
15.0032:   (MOVE-ROBOT R1 L1 LP) [4.0000]
19.0034:   (GIVE-CUP R1 R2 T2 LP B1 B2) [3.0000]
20.0038:   (MOVE-ROBOT R2 LP L3) [2.0000]
22.0040:   (SERVE-CUP R2 P2 L3 T2 B2) [3.0000]
22.0038:   (MOVE-ROBOT R1 LP L1) [4.0000]
26.0040:   (FILL-CUP R1 M1 L1 T1 B2) [2.0000]
28.0044:   (MOVE-ROBOT R1 L1 L2) [2.0000]
30.0046:   (SERVE-CUP R1 P1 L2 T1 B2) [3.0000]
```

4.4.2. Plan para el recurso no renovable, minimizando la batería y con 3 robots

```
; Time 0.27
; Search time 0.24
; Parsing time 0.02
; Mutex time 0.01
; MetricValue 16.92

0.0002:    (GET-CUP R2 A1 L3 T3 B1) [2.0000]
2.0006:    (GET-CUP R2 A1 L3 T2 B2) [2.0000]
0.0002:    (MOVE-ROBOT R1 L1 LP) [4.0000]
4.0010:    (MOVE-ROBOT R2 L3 LP) [1.3333]
5.3345:    (GIVE-CUP R2 R1 T2 LP B2 B1) [1.0000]
4.0004:    (MOVE-ROBOT R1 LP L1) [4.0000]
8.0006:    (FILL-CUP R1 M1 L1 T2 B1) [2.0000]
10.0010:    (MOVE-ROBOT R1 L1 L2) [2.0000]
12.0012:    (MOVE-ROBOT R1 L2 L1) [2.0000]
14.0014:    (MOVE-ROBOT R1 L1 LP) [4.0000]
18.0016:    (GIVE-CUP R1 R2 T2 LP B1 B2) [3.0000]
19.6687:    (MOVE-ROBOT R2 LP L3) [1.3333]
21.0022:    (SERVE-CUP R2 P2 L3 T2 B2) [3.0000]
24.0024:    (GET-CUP R2 A1 L3 T1 B2) [2.0000]
26.0028:    (MOVE-ROBOT R2 L3 LP) [1.3333]
27.3363:    (GIVE-CUP R2 R1 T1 LP B2 B1) [1.0000]
28.3367:    (MOVE-ROBOT R2 LP L3) [1.3333]
29.6702:    (MOVE-ROBOT R2 L3 LP) [1.3333]
24.3367:    (MOVE-ROBOT R1 LP L1) [4.0000]
28.3369:    (FILL-CUP R1 M1 L1 T1 B1) [2.0000]
30.3373:    (MOVE-ROBOT R1 L1 LP) [4.0000]
31.0037:    (MOVE-ROBOT R2 LP L3) [1.3333]
32.3372:    (MOVE-ROBOT R2 L3 LP) [1.3333]
34.3375:    (GIVE-CUP R2 R1 T3 LP B1 B2) [1.0000]
34.3375:    (MOVE-ROBOT R1 LP L1) [4.0000]
38.3377:    (FILL-CUP R1 M1 L1 T3 B2) [2.0000]
40.3381:    (MOVE-ROBOT R1 L1 L2) [2.0000]
42.3383:    (SERVE-CUP R1 P1 L2 T1 B1) [3.0000]
45.3387:    (MOVE-ROBOT R1 L2 LP) [4.0000]
49.3389:    (GIVE-CUP R1 R2 T3 LP B2 B1) [3.0000]
51.0060:    (MOVE-ROBOT R2 LP L3) [1.3333]
52.3395:    (SERVE-CUP R2 P3 L3 T3 B1) [3.0000]
```

4.4.3. Plan para el recurso renovable, minimizando la batería y con 2 robots

```
; Time 0.06
; Search time 0.04
; Parsing time 0.02
; Mutex time 0.00
; MetricValue 8.50

0.0002:    (GET-CUP R2 A1 L3 T2 B2) [2.0000]
0.0002:    (MOVE-ROBOT R1 L1 LP) [4.0000]
2.0006:    (GET-CUP R2 A1 L3 T1 B1) [2.0000]
4.0010:    (MOVE-ROBOT R2 L3 LP) [2.0000]
6.0012:    (GIVE-CUP R2 R1 T1 LP B1 B2) [1.0000]
7.0016:    (MOVE-ROBOT R2 LP L3) [2.0000]
9.0018:    (MOVE-ROBOT R2 L3 LP) [2.0000]
11.0020:    (GIVE-CUP R2 R1 T2 LP B2 B1) [1.0000]
8.0024:    (MOVE-ROBOT R1 LP L1) [4.0000]
12.0026:    (FILL-CUP R1 M1 L1 T1 B2) [2.0000]
```

```
14.0030:   (MOVE-ROBOT R1 L1 LP) [4.0000]
18.0032:   (MOVE-ROBOT R1 LP L1) [4.0000]
22.0034:   (FILL-CUP R1 M1 L1 T2 B1) [2.0000]
24.0038:   (MOVE-ROBOT R1 L1 LP) [4.0000]
28.0040:   (GIVE-CUP R1 R2 T2 LP B1 B2) [3.0000]
29.0044:   (MOVE-ROBOT R2 LP L3) [2.0000]
31.0044:   (MOVE-ROBOT R1 LP L2) [4.0000]
35.0046:   (SERVE-CUP R1 P1 L2 T1 B2) [3.0000]
31.0046:   (SERVE-CUP R2 P2 L3 T2 B2) [3.0000]
```

4.4.4. Plan para el recurso renovable, minimizando la batería y con 3 robots

```
; Time 0.05
; Search time 0.05
; Parsing time 0.00
; Mutex time 0.00
; MetricValue 25.08

0.0002:   (GET-CUP R2 A1 L3 T2 B2) [2.0000]
2.0006:   (GET-CUP R2 A1 L3 T1 B1) [2.0000]
0.0002:   (MOVE-ROBOT R1 L1 LP) [4.0000]
4.0008:   (GIVE-CUP R2 R3 T1 L3 B1 B2) [1.0000]
5.0010:   (GIVE-CUP R3 R2 T1 L3 B2 B1) [1.0000]
5.0012:   (MOVE-ROBOT R2 L3 LP) [1.3333]
6.3347:   (GIVE-CUP R2 R1 T1 LP B1 B2) [1.0000]
4.0004:   (MOVE-ROBOT R1 LP L1) [4.0000]
8.0006:   (FILL-CUP R1 M1 L1 T1 B2) [2.0000]
10.0010:   (MOVE-ROBOT R1 L1 L2) [2.0000]
12.0012:   (SERVE-CUP R1 P1 L2 T1 B2) [3.0000]
15.0016:   (MOVE-ROBOT R1 L2 LP) [4.0000]
7.3351:   (MOVE-ROBOT R2 LP L3) [1.3333]
8.6686:   (GET-CUP R2 A1 L3 T3 B1) [2.0000]
10.6688:   (GIVE-CUP R2 R3 T3 L3 B1 B2) [1.0000]
11.6690:   (GIVE-CUP R3 R2 T3 L3 B2 B1) [1.0000]
12.6692:   (GIVE-CUP R2 R3 T3 L3 B1 B2) [1.0000]
13.6694:   (GIVE-CUP R3 R2 T3 L3 B2 B1) [1.0000]
14.6696:   (GIVE-CUP R2 R3 T3 L3 B1 B2) [1.0000]
15.6698:   (GIVE-CUP R3 R2 T3 L3 B2 B1) [1.0000]
15.6700:   (MOVE-ROBOT R2 L3 LP) [1.3333]
19.0018:   (GIVE-CUP R2 R1 T3 LP B1 B2) [1.0000]
19.0018:   (MOVE-ROBOT R1 LP L1) [4.0000]
23.0020:   (FILL-CUP R1 M1 L1 T3 B2) [2.0000]
25.0024:   (MOVE-ROBOT R1 L1 LP) [4.0000]
29.0026:   (GIVE-CUP R1 R2 T3 LP B2 B1) [3.0000]
32.0030:   (MOVE-ROBOT R1 LP L1) [4.0000]
30.6697:   (MOVE-ROBOT R2 LP L3) [1.3333]
32.0032:   (MOVE-ROBOT R2 L3 LP) [1.3333]
33.3367:   (MOVE-ROBOT R2 LP L3) [1.3333]
34.6702:   (GIVE-CUP R2 R3 T3 L3 B1 B2) [3.0000]
37.6704:   (GIVE-CUP R3 R2 T3 L3 B2 B1) [3.0000]
40.6706:   (SERVE-CUP R2 P3 L3 T3 B1) [3.0000]
43.6710:   (MOVE-ROBOT R2 L3 LP) [1.3333]
36.0032:   (MOVE-ROBOT R1 L1 LP) [4.0000]
45.0045:   (GIVE-CUP R2 R1 T2 LP B2 B1) [1.0000]
42.0049:   (MOVE-ROBOT R1 LP L1) [4.0000]
46.0051:   (FILL-CUP R1 M1 L1 T2 B1) [2.0000]
48.0055:   (MOVE-ROBOT R1 L1 LP) [4.0000]
52.0057:   (GIVE-CUP R1 R2 T2 LP B1 B2) [3.0000]
53.6728:   (MOVE-ROBOT R2 LP L3) [1.3333]
55.0063:   (SERVE-CUP R2 P2 L3 T2 B2) [3.0000]
```

4.4.5. Plan para el recurso no renovable, minimizando el tiempo y con 2 robots

```
; Time 0.04
; Search time 0.03
; Parsing time 0.01
; Mutex time 0.00
; MakeSpan 28.00


0.0002:    (GET-CUP R2 A1 L3 T1 B1) [2.0000]
2.0006:    (GET-CUP R2 A1 L3 T2 B2) [2.0000]
4.0010:    (MOVE-ROBOT R2 L3 LP) [2.0000]
0.0002:    (MOVE-ROBOT R1 L1 LP) [4.0000]
6.0012:    (GIVE-CUP R2 R1 T2 LP B2 B1) [1.0000]
4.0004:    (MOVE-ROBOT R1 LP L1) [4.0000]
8.0006:    (FILL-CUP R1 M1 L1 T2 B1) [2.0000]
10.0010:    (MOVE-ROBOT R1 L1 LP) [4.0000]
7.0016:    (MOVE-ROBOT R2 LP L3) [2.0000]
9.0018:    (MOVE-ROBOT R2 L3 LP) [2.0000]
14.0012:    (GIVE-CUP R2 R1 T1 LP B1 B2) [1.0000]
14.0012:    (GIVE-CUP R1 R2 T2 LP B1 B2) [3.0000]
17.0016:    (MOVE-ROBOT R1 LP L1) [4.0000]
21.0018:    (FILL-CUP R1 M1 L1 T1 B2) [2.0000]
23.0022:    (MOVE-ROBOT R1 L1 L2) [2.0000]
25.0024:    (SERVE-CUP R1 P1 L2 T1 B2) [3.0000]
15.0016:    (MOVE-ROBOT R2 LP L3) [2.0000]
17.0018:    (SERVE-CUP R2 P2 L3 T2 B2) [3.0000]
```

4.4.6. Plan para el recurso no renovable, minimizando el tiempo y con 3 robots

```
; Time 0.08
; Search time 0.08
; Parsing time 0.00
; Mutex time 0.00
; MakeSpan 290.00


0.0002:    (GET-CUP R2 A1 L3 T3 B1) [2.0000]
2.0006:    (GET-CUP R2 A1 L3 T2 B2) [2.0000]
0.0002:    (MOVE-ROBOT R1 L1 LP) [4.0000]
4.0010:    (MOVE-ROBOT R2 L3 LP) [1.3333]
5.3345:    (GIVE-CUP R2 R1 T2 LP B2 B1) [1.0000]
4.0004:    (MOVE-ROBOT R1 LP L1) [4.0000]
8.0006:    (FILL-CUP R1 M1 L1 T2 B1) [2.0000]
10.0010:    (MOVE-ROBOT R1 L1 LP) [4.0000]
14.0012:    (GIVE-CUP R1 R2 T2 LP B1 B2) [3.0000]
15.6683:    (MOVE-ROBOT R2 LP L3) [1.3333]
17.0018:    (GIVE-CUP R2 R3 T2 L3 B2 B1) [3.0000]
20.0020:    (GET-CUP R2 A1 L3 T1 B2) [2.0000]
22.0024:    (MOVE-ROBOT R2 L3 LP) [1.3333]
17.0016:    (MOVE-ROBOT R1 LP L1) [4.0000]
21.0018:    (MOVE-ROBOT R1 L1 LP) [4.0000]
25.0020:    (MOVE-ROBOT R1 LP L2) [4.0000]
29.0022:    (MOVE-ROBOT R1 L2 LP) [4.0000]
33.0024:    (MOVE-ROBOT R1 LP L2) [4.0000]
37.0026:    (MOVE-ROBOT R1 L2 LP) [4.0000]
41.0028:    (MOVE-ROBOT R1 LP L2) [4.0000]
45.0030:    (MOVE-ROBOT R1 L2 LP) [4.0000]
49.0032:    (MOVE-ROBOT R1 LP L2) [4.0000]
```

```
53.0034:   (MOVE-ROBOT R1 L2 LP) [4.0000]
57.0036:   (MOVE-ROBOT R1 LP L2) [4.0000]
61.0038:   (MOVE-ROBOT R1 L2 LP) [4.0000]
65.0040:   (MOVE-ROBOT R1 LP L2) [4.0000]
69.0042:   (MOVE-ROBOT R1 L2 LP) [4.0000]
73.0044:   (MOVE-ROBOT R1 LP L1) [4.0000]
77.0046:   (MOVE-ROBOT R1 L1 L2) [2.0000]
79.0048:   (MOVE-ROBOT R1 L2 L1) [2.0000]
81.0050:   (MOVE-ROBOT R1 L1 LP) [4.0000]
85.0052:   (MOVE-ROBOT R1 LP L1) [4.0000]
89.0054:   (MOVE-ROBOT R1 L1 L2) [2.0000]
91.0056:   (MOVE-ROBOT R1 L2 LP) [4.0000]
95.0058:   (MOVE-ROBOT R1 LP L2) [4.0000]
99.0060:   (MOVE-ROBOT R1 L2 LP) [4.0000]
103.0062:   (MOVE-ROBOT R1 LP L2) [4.0000]
107.0064:   (MOVE-ROBOT R1 L2 LP) [4.0000]
111.0066:   (MOVE-ROBOT R1 LP L2) [4.0000]
115.0068:   (MOVE-ROBOT R1 L2 LP) [4.0000]
119.0070:   (MOVE-ROBOT R1 LP L2) [4.0000]
123.0072:   (MOVE-ROBOT R1 L2 LP) [4.0000]
127.0074:   (MOVE-ROBOT R1 LP L2) [4.0000]
131.0076:   (MOVE-ROBOT R1 L2 LP) [4.0000]
135.0078:   (MOVE-ROBOT R1 LP L2) [4.0000]
139.0079:   (MOVE-ROBOT R1 L2 LP) [4.0000]
143.0081:   (MOVE-ROBOT R1 LP L2) [4.0000]
147.0083:   (MOVE-ROBOT R1 L2 LP) [4.0000]
151.0085:   (MOVE-ROBOT R1 LP L2) [4.0000]
155.0087:   (MOVE-ROBOT R1 L2 LP) [4.0000]
159.0089:   (MOVE-ROBOT R1 LP L2) [4.0000]
163.0091:   (MOVE-ROBOT R1 L2 LP) [4.0000]
167.0093:   (MOVE-ROBOT R1 LP L2) [4.0000]
171.0095:   (MOVE-ROBOT R1 L2 LP) [4.0000]
175.0097:   (MOVE-ROBOT R1 LP L2) [4.0000]
179.0099:   (MOVE-ROBOT R1 L2 LP) [4.0000]
183.0101:   (MOVE-ROBOT R1 LP L2) [4.0000]
187.0103:   (MOVE-ROBOT R1 L2 LP) [4.0000]
191.0105:   (MOVE-ROBOT R1 LP L2) [4.0000]
195.0107:   (MOVE-ROBOT R1 L2 LP) [4.0000]
199.0109:   (MOVE-ROBOT R1 LP L2) [4.0000]
203.0111:   (MOVE-ROBOT R1 L2 LP) [4.0000]
207.0113:   (MOVE-ROBOT R1 LP L2) [4.0000]
211.0115:   (MOVE-ROBOT R1 L2 LP) [4.0000]
215.0117:   (MOVE-ROBOT R1 LP L2) [4.0000]
219.0119:   (MOVE-ROBOT R1 L2 LP) [4.0000]
223.0121:   (MOVE-ROBOT R1 LP L2) [4.0000]
227.0123:   (MOVE-ROBOT R1 L2 LP) [4.0000]
231.0125:   (MOVE-ROBOT R1 LP L2) [4.0000]
235.0127:   (MOVE-ROBOT R1 L2 LP) [4.0000]
239.0129:   (MOVE-ROBOT R1 LP L2) [4.0000]
243.0131:   (MOVE-ROBOT R1 L2 LP) [4.0000]
247.0133:   (MOVE-ROBOT R1 LP L2) [4.0000]
251.0135:   (MOVE-ROBOT R1 L2 LP) [4.0000]
255.0137:   (GIVE-CUP R2 R1 T1 LP B2 B1) [1.0000]
255.0137:   (MOVE-ROBOT R1 LP L1) [4.0000]
259.0139:   (FILL-CUP R1 M1 L1 T1 B1) [2.0000]
261.0143:   (MOVE-ROBOT R1 L1 L2) [2.0000]
263.0146:   (SERVE-CUP R1 P1 L2 T1 B1) [3.0000]
266.0150:   (MOVE-ROBOT R1 L2 LP) [4.0000]
256.0141:   (MOVE-ROBOT R2 LP L4) [1.3333]
257.3477:   (MOVE-ROBOT R2 L4 L3) [0.6667]
258.0146:   (GIVE-CUP R3 R2 T2 L3 B1 B2) [3.0000]
261.0148:   (SERVE-CUP R2 P2 L3 T2 B2) [3.0000]
264.0152:   (MOVE-ROBOT R2 L3 LP) [1.3333]
270.0152:   (GIVE-CUP R2 R1 T3 LP B1 B2) [1.0000]
```

```
271.0154:   (GIVE-CUP R1 R2 T3 LP B2 B1) [1.0000]
272.0156:   (GIVE-CUP R2 R1 T3 LP B1 B2) [1.0000]
273.0158:   (GIVE-CUP R1 R2 T3 LP B2 B1) [1.0000]
274.0161:   (GIVE-CUP R2 R1 T3 LP B1 B2) [1.0000]
274.0163:   (MOVE-ROBOT R1 LP L1) [4.0000]
278.0165:   (FILL-CUP R1 M1 L1 T3 B2) [2.0000]
280.0169:   (MOVE-ROBOT R1 L1 LP) [4.0000]
284.0171:   (GIVE-CUP R1 R2 T3 LP B2 B1) [3.0000]
285.6842:   (MOVE-ROBOT R2 LP L3) [1.3333]
287.0178:   (SERVE-CUP R2 P3 L3 T3 B1) [3.0000]
```

4.4.7. Plan para el recurso renovable, minimizando el tiempo y con 2 robots

```
; Time 0.05
; Search time 0.05
; Parsing time 0.00
; Mutex time 0.00
; MakeSpan 39.00


0.0002:   (GET-CUP R2 A1 L3 T2 B1) [2.0000]
0.0002:   (MOVE-ROBOT R1 L1 LP) [4.0000]
2.0006:   (GET-CUP R2 A1 L3 T1 B2) [2.0000]
4.0010:   (MOVE-ROBOT R2 L3 LP) [2.0000]
6.0012:   (GIVE-CUP R2 R1 T1 LP B2 B1) [1.0000]
7.0016:   (MOVE-ROBOT R2 LP L3) [2.0000]
9.0018:   (MOVE-ROBOT R2 L3 LP) [2.0000]
11.0020:   (GIVE-CUP R2 R1 T2 LP B1 B2) [1.0000]
8.0024:   (MOVE-ROBOT R1 LP L1) [4.0000]
12.0026:   (FILL-CUP R1 M1 L1 T1 B1) [2.0000]
14.0030:   (MOVE-ROBOT R1 L1 LP) [4.0000]
18.0032:   (MOVE-ROBOT R1 LP L1) [4.0000]
22.0034:   (FILL-CUP R1 M1 L1 T2 B2) [2.0000]
24.0038:   (MOVE-ROBOT R1 L1 L2) [2.0000]
26.0040:   (SERVE-CUP R1 P1 L2 T1 B1) [3.0000]
29.0044:   (MOVE-ROBOT R1 L2 LP) [4.0000]
33.0046:   (GIVE-CUP R1 R2 T2 LP B2 B1) [3.0000]
34.0050:   (MOVE-ROBOT R2 LP L3) [2.0000]
36.0052:   (SERVE-CUP R2 P2 L3 T2 B1) [3.0000]
```

4.4.8. Plan para el recurso renovable, minimizando el tiempo y con 3 robots

```
; Time 0.07
; Search time 0.05
; Parsing time 0.01
; Mutex time 0.01
; MakeSpan 65.00


0.0002:   (GET-CUP R2 A1 L3 T3 B2) [2.0000]
0.0002:   (MOVE-ROBOT R1 L1 LP) [4.0000]
2.0004:   (GIVE-CUP R2 R3 T3 L3 B2 B1) [1.0000]
3.0006:   (GIVE-CUP R3 R2 T3 L3 B1 B2) [1.0000]
4.0008:   (GIVE-CUP R2 R3 T3 L3 B2 B1) [1.0000]
5.0010:   (GIVE-CUP R3 R2 T3 L3 B1 B2) [1.0000]
5.0012:   (MOVE-ROBOT R2 L3 LP) [1.3333]
6.3347:   (GIVE-CUP R2 R1 T3 LP B2 B1) [1.0000]
```

```
16 7.3351:   (MOVE-ROBOT R2 LP L3) [1.3333]
17 8.6686:   (MOVE-ROBOT R2 L3 LP) [1.3333]
18 10.0021:   (MOVE-ROBOT R2 LP L3) [1.3333]
19 11.3356:   (MOVE-ROBOT R2 L3 LP) [1.3333]
20 12.6691:   (MOVE-ROBOT R2 LP L3) [1.3333]
21 14.0026:   (MOVE-ROBOT R2 L3 LP) [1.3333]
22 15.3361:   (MOVE-ROBOT R2 LP L3) [1.3333]
23 16.6696:   (MOVE-ROBOT R2 L3 LP) [1.3333]
24 18.0031:   (MOVE-ROBOT R2 LP L3) [1.3333]
25 19.3366:   (GET-CUP R2 A1 L3 T1 B1) [2.0000]
26 4.0004:   (MOVE-ROBOT R1 LP L1) [4.0000]
27 8.0006:   (MOVE-ROBOT R1 L1 LP) [4.0000]
28 21.3370:   (MOVE-ROBOT R2 L3 LP) [1.3333]
29 12.0008:   (MOVE-ROBOT R1 LP L1) [4.0000]
30 16.0010:   (FILL-CUP R1 M1 L1 T3 B1) [2.0000]
31 18.0014:   (MOVE-ROBOT R1 L1 LP) [4.0000]
32 22.6705:   (GIVE-CUP R2 R1 T1 LP B1 B2) [1.0000]
33 22.0016:   (MOVE-ROBOT R1 LP L1) [4.0000]
34 26.0018:   (FILL-CUP R1 M1 L1 T1 B2) [2.0000]
35 28.0022:   (MOVE-ROBOT R1 L1 LP) [4.0000]
36 32.0024:   (MOVE-ROBOT R1 LP L2) [4.0000]
37 36.0026:   (SERVE-CUP R1 P1 L2 T1 B2) [3.0000]
38 39.0030:   (MOVE-ROBOT R1 L2 LP) [4.0000]
39 6.0014:   (GET-CUP R3 A1 L3 T2 B2) [2.0000]
40 23.6709:   (MOVE-ROBOT R2 LP L3) [1.3333]
41 25.0044:   (GIVE-CUP R3 R2 T2 L3 B2 B1) [1.0000]
42 25.0044:   (MOVE-ROBOT R2 L3 LP) [1.3333]
43 43.0032:   (GIVE-CUP R2 R1 T2 LP B1 B2) [1.0000]
44 43.0032:   (MOVE-ROBOT R1 LP L1) [4.0000]
45 47.0034:   (FILL-CUP R1 M1 L1 T2 B2) [2.0000]
46 49.0038:   (MOVE-ROBOT R1 L1 LP) [4.0000]
47 53.0040:   (GIVE-CUP R1 R2 T2 LP B2 B1) [3.0000]
48 56.0044:   (GIVE-CUP R1 R2 T3 LP B1 B2) [3.0000]
49 57.6715:   (MOVE-ROBOT R2 LP L3) [1.3333]
50 59.0050:   (SERVE-CUP R2 P2 L3 T2 B1) [3.0000]
51 62.0054:   (SERVE-CUP R2 P3 L3 T3 B2) [3.0000]
```

En vista de los resultados obtenidos, podemos afirmar que el planificador LPG resuelve de manera satisfactoria los problemas propuestos, pero que en algunas ocasiones, dependiendo de la complejidad del mismo es incapaz de dar una solución ya que está fuera de su alcance. También hemos podido observar ciertas incoherencias, dependiendo del recurso que se desee minimizar, ya que hay veces que un robot se dirige a una habitación para inmediatamente volver a la habitación anterior. Si se minimiza batería le da igual perder tiempo realizando dicha acción innecesaria.

Renovable de 3 robots minimizando batería obtiene peores resultados que el no renovable. Consideramos que esto puede deberse a que en la situación de no renovable uno de los robots se quedó sin batería. Por lo tanto, no solamente no consume (quedarían solamente 2 robots consumiendo batería), sino que puede que además esos dos robots restantes trabajaran de una forma mucho más eficiente.

Además, con respecto al numero de pasos obtenidos, observamos que el dominio numérico obtiene unos resultados mucho mejores que el proposicional y el temporal, salvo cuando definimos un problema con 3 robots, minimizando tiempo y con recurso no renovable, cuyo resultado fue un plan demasiado largo lleno de incoherencias, por lo que podemos afirmar que en ciertos planes complejos LPG no responde bien.

En la figura 4.1, se pueden apreciar los resultados obtenidos en forma de gráfica. En esta, la duración representa el *MakeSpan* (duración de las acciones) cuando la métrica a minimizar es el tiempo; y el *MetricValue* cuando la métrica a minimizar es la batería total gastada.

Figura 4.1: Gráfica comparativa con los resultados de los tres planificadores. No se ha incluido el caso de minimizar tiempo con recurso no renovable y 3 robots porque el coste era mucho mayor que en los otros casos.

CAPÍTULO 5
Árbol POP

En este capítulo se simularán tres iteraciones del algoritmo POP que se utiliza en los planificadores de orden parcial vistos en clase.

5.1 Plan inicial

Para la realización de este apartado, se nos ofrece un problema de complejidad menor al que se ha estado trabajando en los anteriores apartados.

En nuestro caso, el objetivo del problema será que una sola persona (hemos escogido la que se encuentra en la habitación 2) obtenga su taza de té.

Por tanto, los objetivos del estado final para esta instancia serían:

(on t1 p1)(full t1)

El estado inicial contaría con los mismos predicados, pero los más importantes serían:

(on t1 a1)(at r1 l1)(at r2 l3)(at a1 l3)(at p1 l2)(empty t1)

El nodo que representa esta situación inicial se puede apreciar en la figura 5.1

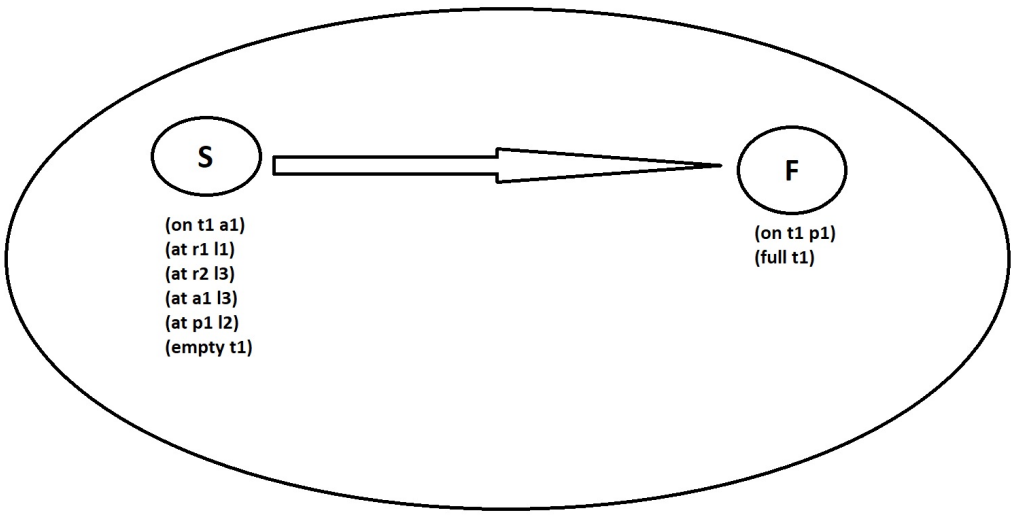

Figura 5.1: Nodo POP del estado inicial.

5.2 Primera iteración

Partiendo del plan inicial, nos encontramos con los siguientes *flaws*:

- **Subobjetivos**: (on t1 p1)(full t1)

Para esta primera iteración, se ha escogido (on t1 p1) como objetivo. Este objetivo puede cumplirse en el caso de que se ejecute la acción *serve-cup*.

El resultado se puede apreciar en la figura 5.3.

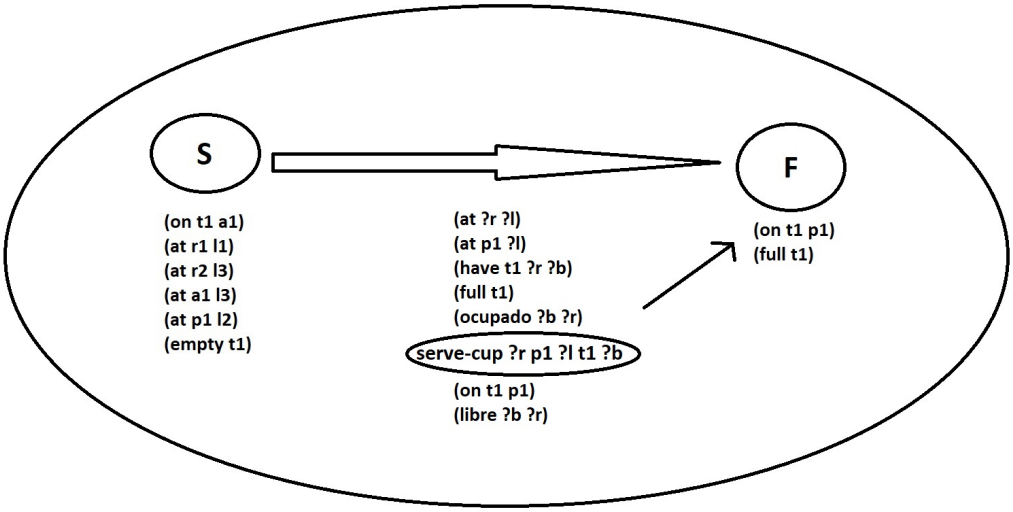

Figura 5.2: Nodo POP de la iteración 1.

La flecha que sale de *serve-cup* indica un enlace causal entre esta y el estado final.

Aquí se pueden apreciar los siguientes *flaws*:

- **Subobjetivos**: (full t1)(at ?r ?l)(at p1 ?l)(have t1 ?r ?b)(ocupado ?b ?r)
- **Variables a instanciar**: ?r, ?l, ?b

5.3 Segunda iteración

Aquí se va a escoger otro subobjetivo a resolver, a falta de alguna amenaza en la acción que por el momento se ha realizado.

El subobjetivo escogido en este caso va a ser (full t1). Éste puede ser resuelto por la acción *fill-cup*.

El resultado se puede apreciar en la figura **??**.

De nuevo, hay otro enlace causal, en este caso entre *fill-cup* y *serve-cup*.

Aquí se pueden apreciar los siguientes *flaws*:

- **Subobjetivos**: (at ?r ?l)(at p1 ?l)(have t1 ?r ?b)(ocupado ?b ?r)(at ?m ?l)(empty t1)
- **Variables a instanciar**: ?r, ?l, ?b, ?m

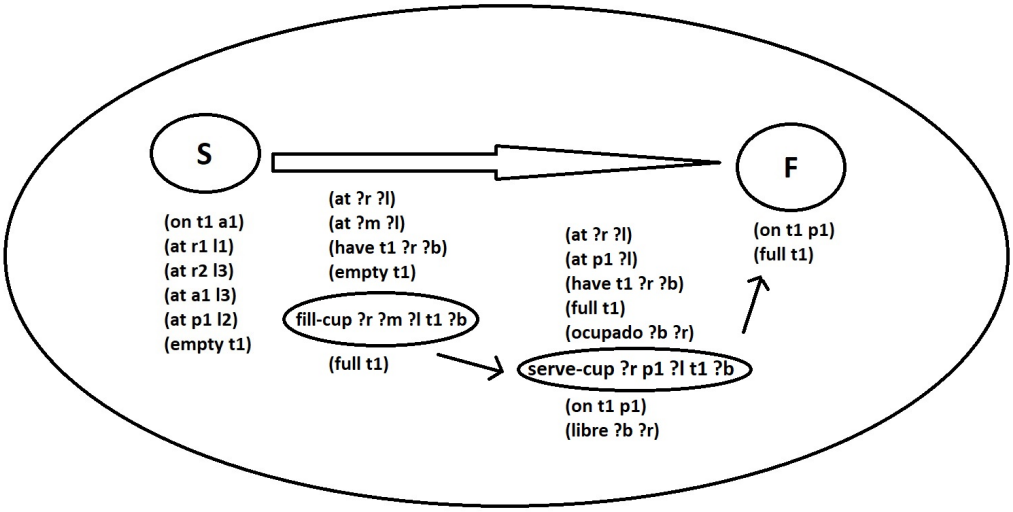

Figura 5.3: Nodo POP de la iteración 2.

5.4 Tercera iteración

Por último, en esta iteración escogeremos algunas variables por instanciar y las resolveremos.

Primero, resolvemos ?r, que puede tomar como valores r1, r2. Como ?r es el que ha de servir la taza al usuario p1 y este se encuentra en la habitación 2, no permitida para el robot 2, entonces ?r = r1.

Por último, resolvemos ?m que, como en nuestro problema solo hay una máquina de hacer té, tomará el valor de m1.

De esta forma, para un nodo sucesor, los *flaws* serán los siguientes:

- **Subobjetivos**: (at r1 ?l)(at p1 ?l)(have t1 r1 ?b)(ocupado ?b r1)(at m1 ?l)(empty t1)
- **Variables a instanciar**: ?l, ?b

CAPÍTULO 6
Graphplan

En este capítulo se pretende ejecutar el algoritmo Graphplan partiendo del problema del primer ejercicio. En este caso, se construirá el grafo de planificación relajado, que no tiene en cuenta los efectos negativos de las acciones ni calcula las relaciones de exclusión mútua.

6.1 Plan inicial

La siguiente imagen muestra el problema sobre el que se trabajará:

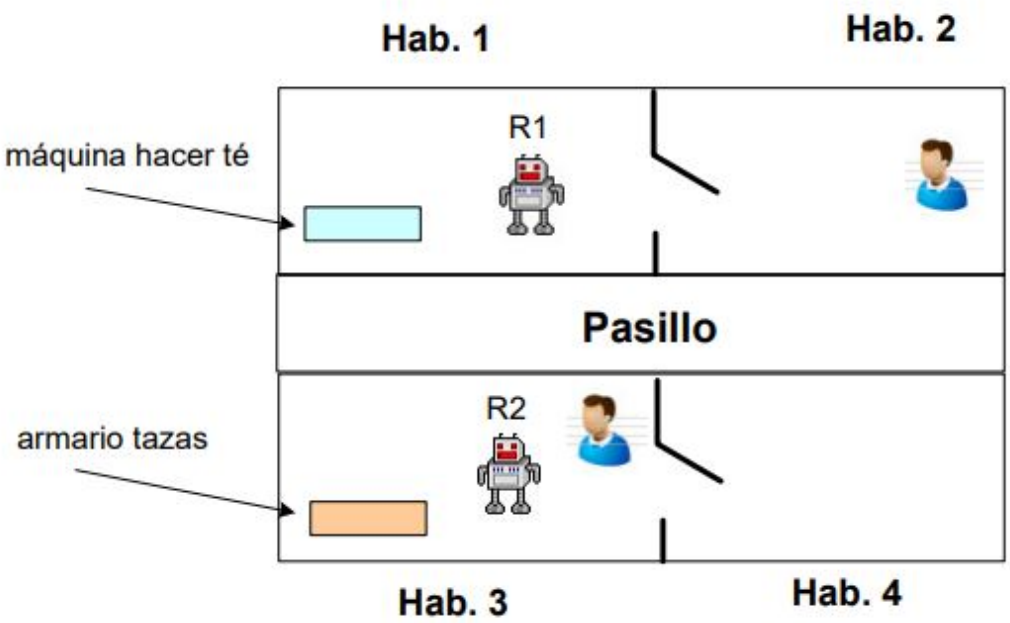

Figura 6.1: Escenario original.

El objetivo del problema sigue siendo el mismo: servir dos tazas llenas (t1 y t2), una a cada uno de los dos usuarios (p1 y p2).

Teniendo en cuenta tanto este escenario como la implementación del dominio en PDDL, contaríamos pues con dos robots (cada uno con dos brazos), dos tazas situadas inicialmente en el armario en la habitación 3 y una máquina de hacer té en la habitación 1.

Los objetivos para este problema son los siguientes:

(on t1 p1)(on t2 p2)(full t1)(full t2)

6.2 Traza del algoritmo

	P(0)	A(1)	
1	(at r1 l1)	move-robot (r1 l1 lp)	P={11, 23}; E={33}
2	(at r2 l3)	move-robot (r2 l3 lp)	P={17, 26}; E={34}
3	(at p1 l2)		
4	(at p2 l3)		
5	(at m1 l1)		
6	(at a1 l3)		
7	(on t1 a1)	get-cup (r2 a1 l3 t1 b1)	P={2, 6, 31}; E={35,36}
8	(on t2 a1)	get-cup (r2 a1 l3 t2 b2)	P={2, 6, 33}; E={37,38}
9	(linked l1 l2)		
10	(linked l2 l1)		
11	(linked l1 lp)		
12	(linked lp l1)		
13	(linked l2 lp)		
14	(linked lp l2)		
15	(linked l3 l4)		
16	(linked l4 l3)		
17	(linked l3 lp)		
18	(linked lp l3)		
19	(linked l4 lp)		
20	(linked lp l4)		
21	(allowed r1 l1)		
22	(allowed r1 l2)		
23	(allowed r1 lp)		
24	(allowed r2 l3)		
25	(allowed r2 l4)		
26	(allowed r2 lp)		
27	(empty t1)		
28	(empty t2)		
29	(libre b1 r1)		
30	(libre b2 r1)		
31	(libre b1 r2)		
32	(libre b2 r2)		
	P(1)	**A(2)**	
33	(at r1 lp)	give-cup (r2 r1 t1 lp b1 b1)	P={34, 35, 29, 36}; E={39, 40,31}
34	(at r2 lp)	give-cup (r2 r1 t2 lp b2 b2)	P={33, 37, 30, 38}; E={41, 42, 32}
35	(have t1 r2 b1)		
36	(ocupado b1 r2)		
37	(have t2 r2 b2)		
38	(ocupado b2 r2)		
	P(2)	**A(3)**	
39	(have t1 r1 b1)	fill-cup(r1 m1 l1 t1 b1)	P={1, 5, 27}; E={43}
40	(ocupado b1 r1)		
41	(have t2 r1 b2)	fill-cup(r1 m1 l1 t2 b2)	P={1, 5, 28}; E={44}
42	(ocupado b2 r1)		
	P(3)	**A(4)**	
43	**(full t1)**	move-robot (r1 l1 l2)	P={1, 9, 22}; E={45}
44	**(full t2)**	serve-cup(r2 p2 l3 t2 b2)	P={2, 4, 37, 38}; E={46, 32}
	P(4)	**A(5)**	
45	(at r1 l2)	serve-cup(r1 p1 l2 t1 b1)	P={3, 39, 43, 40}; E={47, 29}
46	**(on t2 p2)**		
	P(5)		
47	**(on t1 p1)**		

G = (and (on t1 p1) (on t2 p2) (full t1) (full t2))

h_sum(G) = 3 + 3 + 4 + 5 = 15

h_max(G) = 5

plan_relajado(G) = 11

En este caso, el plan se podría sacar sin necesidad de backtracking cogiendo la situación actual como estado inicial, y aplicando todas las posibles acciones que se puedan ejecutar a partir de aquí. De este modo, obtendríamos un árbol de ejecuciones con todas las posibilidades hasta llegar a un estado que a lo largo de su ejecución haya cumplido todos los objetivos.

Al tratarse de una ejecución que combina todas las posibilidades (y su coste varía en función a dichas posibilidades), el tiempo de la ejecución sería polinómico.

En este problema, la heurística más informada es la **h_sum**, dado que es la que más se aproxima al número de pasos que se han realizado realmente, teniendo en cuenta que un paso equivale a un conjunto de acciones que se han realizado en un mismo periodo de tiempo.

Esto se debe a que, pese a ser el grafo de planificación relajado, el estado final con todas las condiciones finales cumpliéndose de forma simultánea se obtendría con el mismo número de pasos (recordemos que en Optic obteníamos un plan de 15 pasos).