Planificación Inteligente : Ascensores

Yuri Vladimir Huallpa Vargas, Florent Covet

Febrero 2023, MIARFID, UPV

Introducción al dominio de ascensores

El dominio de ascensores consiste en llevar un grupo de personas, distribuidos por las plantas de un edificio de 13 plantas hacia otras plantas destino, para ello se cuenta de un conjunto de 5 ascensores; 4 lentos con capacidad de 2 personas y 1 rápido con capacidad de 3 personas. El edificio se separa en 3 bloques, donde el primer bloque cuenta con un ascensor lento que va de la planta 0 hasta la 4; el segundo cuenta con 2 ascensores lentos que va de la planta 4 hasta la 8 y el tercer bloque cuenta con 1 ascensor lento que va del 8 al 12. Además, el ascensor restante, rápido, recorre todo el edificio y únicamente puede detenerse en las plantas pares (0, 2, 4, 6, 8, 10 y 12). Una descripción de todo lo mencionado se puede ver en la Figura 1.

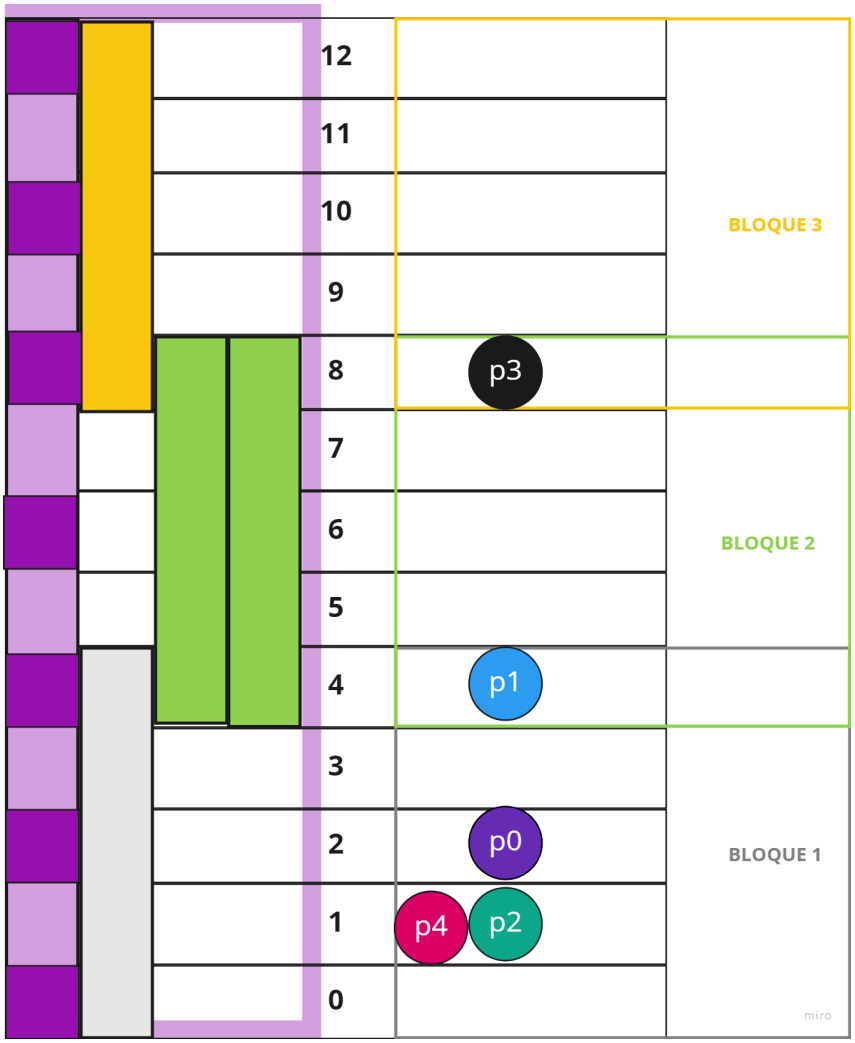


Figura 1. Dominio de ascensores

1. Dominio proposicional

Primero se describirán los predicados y se concluirá con las acciones correspondientes a este dominio.

1.1. Diseño de los predicados

- **(next ?x - integer ?y - integer)**
  El predicado next es un contador, nos ayuda a mantener una relación de secuencia numérica cuando una persona se sube o baja de un ascensor.

- **(above ?x - integer ?y - integer)**
  Above permite identificar el orden de las plantas del edificio, indica si efectivamente una planta está encima de otra.

- **(can-go ?x - lift ?y - integer)**
  El predicado can-go es una restricción, indica las plantas a las cuales un ascensor puede lograr alcanzar.

- **(can-hold ?x - lift ?y - integer)**
  Can-hold nos permite saber si un ascensor aún no alcanzó su capacidad máxima.

- **(passenger-at ?x - passenger ?y - integer)**
  El predicado passenger-at indica si una persona **x** se encuentra en la planta **y** del edificio.

- **(lift-at ?x - lift ?y - integer)**
  Lift-at indica si efectivamente un ascensor **x** está en la planta **y** del edificio.

- **(boarded ?y - passenger ?x - lift)**
  El predicado boarded permite conocer si una persona realmente está dentro del ascensor.

- **(on-board ?x - lift ?y - integer)**
  On-board no permite saber la cantidad real de personas que ya están dentro del ascensor.

1.2. Diseño de los operadores

A. Subir al ascensor (get-on)

Este operador permite que una persona aborde un ascensor cuando este se encuentra en su piso y aún no ha alcanzado su capacidad máxima. Los efectos que produce son que la persona se encuentra dentro del ascensor, ya no está en el piso y reduce la capacidad del ascensor.

```
(:action get-on
         :parameters (?l - lift ?p - passenger ?f - integer ?bn - integer ?an - integer)

         :precondition (and (passenger-at ?p ?f) (lift-at ?l ?f) (on-board ?l ?bn) (can-hold ?l ?an)  (next ?bn ?an))
         :effect
         (and (not (passenger-at ?p ?f)) (boarded ?p ?l)(not (on-board ?l ?bn) ) (on-board ?l ?an)))
```

Figura 1.1. Operador get-on

B. Bajar del ascensor (get-down)

Por otro lado, este operador realiza la acción de bajar a una persona desde el ascensor en una planta del edificio. Los efectos que produce son que la persona abandona el ascensor, se incrementa la capacidad del ascensor y la persona llega a su destino en la planta correspondiente del edificio.

```
(:action get-down
         :parameters (?l - lift ?p - passenger ?f - integer ?bn - integer ?an - integer)
         :precondition (and  (lift-at ?l ?f) (boarded ?p ?l) (on-board ?l ?bn) (next?an ?bn))
         :effect
         (and (not(boarded ?p ?l)) (passenger-at ?p ?f)(not (on-board ?l ?bn) ) (on-board ?l ?an)))
```

Figura 1.2. Operador get-down

C. Viajar hacia una planta superior (move-up)

El operador move-up es el encargado de mover los ascensores de una planta inferior hacia otra superior, siempre que las dos plantas estén dentro de su alcance y que en realidad la planta destino se encuentre por encima de la inicial.

```
(:action move-up
      :parameters (?l - lift ?f1 - integer ?f2 - integer)
      :precondition (and (lift-at ?l ?f1) (can-go ?l ?f2) (above ?f1 ?f2))
      :effect
      (and (not (lift-at ?l ?f1)) (lift-at ?l ?f2) ))
```

Figura 1.3. Operador move-up

D. Viajar hacia una planta inferior (move-down)

El operador move-down es el encargado de mover los ascensores de una planta superior a una inferior, siempre que las dos plantas estén dentro de su alcance y que la planta inicial esté encima de la planta destino.

```
(:action move-down
            :parameters (?l - lift ?f1 - integer ?f2 - integer)
            :precondition (and (lift-at ?l ?f1) (can-go ?l ?f2) (above ?f2 ?f1))
            :effect
     (and (not (lift-at ?l ?f1)) (lift-at ?l ?f2) ))
```

Figura 1.4. Operador move-down

1.3. Definición del problema

Antes de abordar la solución del problema, se dará una notación a cada ascensor. El ascensor en el bloque 1 se llamará "ascensor 1", los dos ascensores en el bloque 2 serán "ascensor 2" y "ascensor 3", el ascensor en el bloque 3 será "ascensor 4" y el ascensor rápido será "ascensor 5".

Dicho esto, el problema inicial se presenta en la Figura 1, a partir de ahora se denominará como problema base y se busca encontrar un plan que nos permita llevar al pasajero p0 a la planta 3, al pasajero p1 hacia la planta 11, al pasajero p2 hacia la planta 12, al pasajero p3 hacia la planta 1 y finalmente al pasajero p4 hacia la planta 9. Los literales y el objetivo se encuentran especificados en el archivo de problemas "ejercicio_1/problem.pddl".

1.4. Experimentacion

Los resultados obtenidos se encontraron después de una serie de experimentos realizados con los planificadores FF, LPG, LPG con la opción timesteps y Optic. Se observó que todos los planificadores produjeron un plan correcto que soluciona el problema. Los resultados se presentarán en el siguiente orden: primero se mostrará la solución del problema base, seguida de las soluciones de las diferentes instancias del problema.

En la Figura 1.5 se observa el plan obtenido por el planificador FF para el problema base, primero los ascensores 5, 2, 1 empiezan a moverse debido a que en las posiciones donde se encontraban no había personas esperando, seguidamente los pasajeros p4 y p2 abordan el ascensor 1 en la planta 1 y posteriormente este sube hacia la siguiente planta a dejar a la persona p4, después las personas p1 y p3 abordan los ascensores 2 y 5 en las plantas 4 y 8. Se sigue así este proceso hasta que el último en llegar a su destino es el pasajero 4.

```
    ff: found legal plan as follows
step    0: MOVE-UP LIFT5 N0 N8
        1: MOVE-DOWN LIFT2 N5 N4
        2: MOVE-UP LIFT1 N0 N1
        3: GET-ON LIFT1 P4 N1 N0 N1
        4: GET-ON LIFT1 P2 N1 N1 N2
        5: MOVE-UP LIFT1 N1 N2
        6: GET-DOWN LIFT1 P4 N2 N2 N1
        7: GET-ON LIFT2 P1 N4 N0 N1
        8: GET-ON LIFT5 P3 N8 N0 N1
        9: MOVE-DOWN LIFT5 N8 N2
       10: MOVE-UP LIFT2 N4 N8
       11: GET-ON LIFT5 P4 N2 N1 N2
       12: GET-DOWN LIFT2 P1 N8 N1 N0
       13: GET-DOWN LIFT5 P3 N2 N2 N1
       14: MOVE-UP LIFT5 N2 N8
       15: GET-DOWN LIFT1 P2 N2 N1 N0
       16: GET-ON LIFT1 P3 N2 N0 N1
       17: GET-DOWN LIFT5 P4 N8 N1 N0
       18: MOVE-DOWN LIFT5 N8 N2
       19: GET-ON LIFT1 P0 N2 N1 N2
       20: MOVE-UP LIFT1 N2 N3
       21: GET-ON LIFT5 P2 N2 N0 N1
       22: MOVE-UP LIFT5 N2 N12
       23: GET-DOWN LIFT5 P2 N12 N1 N0
       24: MOVE-DOWN LIFT4 N9 N8
       25: GET-ON LIFT4 P1 N8 N0 N1
       26: GET-ON LIFT4 P4 N8 N1 N2
       27: MOVE-UP LIFT4 N8 N11
       28: MOVE-DOWN LIFT1 N3 N1
       29: GET-DOWN LIFT1 P3 N1 N2 N1
       30: MOVE-UP LIFT1 N1 N3
       31: GET-DOWN LIFT1 P0 N3 N1 N0
       32: GET-DOWN LIFT4 P1 N11 N2 N1
       33: MOVE-DOWN LIFT4 N11 N9
       34: GET-DOWN LIFT4 P4 N9 N1 N0
```

Figura 1.5. Resultados del plan obtenido por el planificador FF en el problema base.

En cuanto a los resultados de los planificadores, se observa que tanto el planificador FF como el LPG-timesteps logran un plan con menos acciones, pero el planificador OPTIC realiza la tarea en menos pasos, es decir, paraleliza mejor el problema. Se observó que OPTIC utiliza ambos ascensores del bloque 2, a diferencia de LPG que solo hace uso del ascensor 2 y omite en mayor medida el ascensor 3.

Es necesario mencionar que las soluciones que aquí se muestran para el planificador LPG se obtuvieron con 3 iteraciones y que pudo haber encontrado un plan en menos paso si se le hubiera permitido ejecutar con más iteraciones.

Planificador	Acciones	Pasos	Tiempo total
FF	35	35	0.02
LPG	39	27	11.64
LPG - timesteps	35	22	7.91
OPTIC	39	13	0.27

Tabla 1.1. Resultados del plan obtenido por los planificadores FF, LPG, LPG-timesteps y OPTIC en el problema base.

Para concluir, se presentan los resultados de los planes generados por los cuatro planificadores en los experimentos realizados. Donde, el planificador OPTIC ha proporcionado los mejores planes.

Cambiando el estado inicial de los pasajeros			
Planificador	Acciones	Pasos	Tiempo total
FF	30	30	0.02
LPG	32	23	0.10
LPG - timesteps	33	19	0.54
OPTIC	25	**11**	0.07

Tabla 1.2. Resultados del experimento 1

Cambiando el estado inicial y los objetivos			
Planificador	Acciones	Pasos	Tiempo total
FF	26	26	0.01
LPG	40	31	2.34

LPG - timesteps	37	21	1.69
OPTIC	30	**12**	0.66

Tabla 1.3. Resultados del experimento 2

Incrementando el número de pasajeros			
Planificador	Acciones	Pasos	Tiempo total
FF	50	50	0.03
LPG	58	28	3.02
LPG - timesteps	63	29	14.86
OPTIC	52	**24**	0.55

Tabla 1.4. Resultados del experimento 3

Agregando un ascensor lento N6 y uno rápido N7			
Planificador	Acciones	Pasos	Tiempo total
FF	35	35	0.03
LPG	35	19	1.05
LPG - timesteps	44	21	1.35
OPTIC	34	**12**	0.21

Tabla 1.5. Resultados del experimento 4

2. Dominio temporal

En este apartado se nos pide considerar el tiempo de cada acción, para ello se nos proporciona un conjunto de restricciones:

A. Ascensores lentos
   - Subir/bajar una planta: 12 unidades de tiempo
   - Subir/bajar dos plantas: 20 u.t.
   - Subir/bajar tres plantas: 28 u.t.
   - Subir/bajar cuatro plantas: 36 u.t

B. Ascensores rapidos:
   - subir dos plantas: 11 u.t.
   - subir cuatro plantas: 13 u.t.
   - subir seis plantas: 15 u.t.
   - subir ocho plantas: 17 u.t.
   - subir 10 plantas: 19 u.t.
   - subir 12 plantas: 21 u.t.
   - bajar dos plantas: 10 u.t.
   - bajar cuatro plantas: 12 u.t.
   - bajar seis plantas: 14 u.t.
   - bajar ocho subir plantas: 16 u.t.
   - bajar 10 plantas: 18 u.t.
   - bajar 12 plantas: 20 u.t

C. Personas
   - subir/bajar del ascensor: 2 u.t.

Para incorporar estas restricciones se han realizado cambios en el dominio proposicional y en el problema.

2.1 Funciones

Se ha visto adecuado generar dos funciones que nos permitan conocer el tiempo que tarda un ascensor en desplazarse de un punto hacia otro, uno para los ascensores lentos y otro para el rápido.

```
(:functions
        (travel-slow ?f1 - integer ?f2 - integer) 2 - number
        (travel-fast ?f1 - integer ?f2 - integer) 2 - number
)
```

Figura 2.1. Funciones travel-slow y travel fast.

2.2. Operadores

2.2.1. Subir y Bajar del ascensor

Para cumplir el requisito C, se ha modificado los predicados ***get-on*** y ***get-down*** asignándoles un una duración de 2.

```
(:durative-action get-on
          :parameters (?l - lift ?p - passenger ?f - integer ?bn - integer ?an - integer)
          :duration (= ?duration 2)
          :condition (and (at start (passenger-at ?p ?f)) (over all (lift-at ?l ?f))
          (at start (on-board ?l ?bn))(at start (can-hold ?l ?an))  (at start (next ?bn ?an)))
          :effect (and (at start (not (passenger-at ?p ?f))) (at end (boarded ?p ?l))
          You, 5 minutes ago | 1 author (You)
          (at start (not (on-board ?l ?bn) )) (at end (on-board ?l ?an))))
```

Figura 2.2. Operador get-on, dominio temporal

2.2.2 Viajar hacia arriba y abajo.

Para satisfacer las restricciones de tiempo se tuvo que modificar los operadores ***move-up*** y ***move-down*** separándolas una para cada tiempo de ascensor y se les ha asignado el tiempo mediante la función ***travel-slow*** y ***travel-fast.*** Se muestran los operadores *move-up-slow* y *move-up-fast* con los cambios correspondientes.

```
(:durative-action move-up-slow
     :parameters (?l - lift-slow ?f1 - integer ?f2 - integer)
     :duration (= ?duration (travel-slow ?f1 ?f2))
     You, 2 weeks ago | 1 author (You)
     :condition (and (at start(lift-at ?l ?f1)) (at start(can-go ?l ?f2)) (at start(above ?f1 ?f2)))
     :effect
     (and (at start (not (lift-at ?l ?f1))) (at end  (lift-at ?l ?f2))))
```

Figura 2.3. Operador move-up-slow

```
(:durative-action move-up-fast
     :parameters (?l - lift-fast ?f1 - integer ?f2 - integer)
     :duration (= ?duration (travel-fast ?f1 ?f2))
     You, 2 weeks ago | 1 author (You)
     :condition (and (at start(lift-at ?l ?f1)) (at start(can-go ?l ?f2)) (at start(above ?f1 ?f2)))
     :effect
     (and (at start (not (lift-at ?l ?f1))) (at end  (lift-at ?l ?f2))))
```

Figura 2.4. Operador move-up-fast

2.3. Cambios en el problema

En el problema se ha instanciado el valor de las funciones y se les ha asignado un valor según las restricciones proporcionadas.

```
(= (travel-slow n0 n1) 12) (= (travel-slow n0 n2) 20) (= (travel-slow n0 n3) 28) (= (travel-slow n0 n4) 36)
(= (travel-slow n1 n2) 12) (= (travel-slow n1 n3) 20) (= (travel-slow n1 n4) 28)
(= (travel-slow n2 n3) 12) (= (travel-slow n2 n4) 20)
(= (travel-slow n3 n4) 12)
(= (travel-slow n4 n5) 12) (= (travel-slow n4 n6) 20) (= (travel-slow n4 n7) 28) (= (travel-slow n4 n8) 36)
(= (travel-slow n5 n6) 12) (= (travel-slow n5 n7) 20) (= (travel-slow n5 n8) 28)
(= (travel-slow n6 n7) 12) (= (travel-slow n6 n8) 20)
(= (travel-slow n7 n8) 12)
(= (travel-slow n8 n9) 12) (= (travel-slow n8 n10) 20) (= (travel-slow n8 n11) 28) (= (travel-slow n8 n12) 36)
(= (travel-slow n9 n10) 12) (= (travel-slow n9 n11) 20) (= (travel-slow n9 n12) 28)
(= (travel-slow n10 n11) 12) (= (travel-slow n10 n12) 20)
(= (travel-slow n11 n12) 12)

;============================tiempo necesario para alcanzar un piso realizado por el ascensor rapido ============
(= (travel-fast n0 n2) 11) (= (travel-fast n0 n4) 13) (= (travel-fast n0 n6) 15) (= (travel-fast n0 n8) 17)
(= (travel-fast n0 n10) 19)(= (travel-fast n0 n12) 21)
(= (travel-fast n2 n4) 11) (= (travel-fast n2 n6) 13) (= (travel-fast n2 n8) 15) (= (travel-fast n2 n10) 17)
(= (travel-fast n2 n12) 19)
(= (travel-fast n4 n6) 11) (= (travel-fast n4 n8) 13) (= (travel-fast n4 n10) 15) (= (travel-fast n4 n12) 17)
(= (travel-fast n6 n8) 11) (= (travel-fast n6 n10) 13) (= (travel-fast n6 n12) 15)
(= (travel-fast n8 n10) 11) (= (travel-fast n8 n12) 13)
(= (travel-fast n10 n12) 11)

(= (travel-fast n12 n10) 10) (= (travel-fast n12 n8) 12) (= (travel-fast n12 n6) 14) (= (travel-fast n12 n4) 16)
(= (travel-fast n12 n2) 18) (= (travel-fast n12 n0) 20)
(= (travel-fast n10 n8) 10) (= (travel-fast n10 n6) 12) (= (travel-fast n10 n4) 14) (= (travel-fast n10 n2) 16)
 (= (travel-fast n10 n0) 18)
(= (travel-fast n8 n6) 10) (= (travel-fast n8 n4) 12) (= (travel-fast n8 n2) 14) (= (travel-fast n8 n0) 16)
(= (travel-fast n6 n4) 10) (= (travel-fast n6 n2) 12) (= (travel-fast n6 n0) 14)
(= (travel-fast n4 n2) 10) (= (travel-fast n4 n0) 12)
(= (travel-fast n2 n0) 10)
```

Figura 2.5. Instancias del las funciones travel-fast y travel-slow

Habiendo realizado estos cambios se ha conseguido cumplir con todas las restricciones solicitadas. Se ha corroborado el correcto funcionamiento realizando un conjunto de pruebas.

2.4. Experimentacion

Se ha lanzado la instancia inicial del problema pero esta vez se ha ejecutado el planificador LPG con 5 iteraciones, se observó que los planificadores devuelven un plan correcto del problema. En la Tabla 2.1 se observa que el planificador LPG realiza el plan con 38 acciones pero le toma más tiempo en realizarlo.

Planificador	Acciones	Duracion	Tiempo
LPG	38	139	6.35
OPTIC	41	**133**	0.88

Tabla 2.1. Resultados del planificador LPG y OPTIC, dominio temporal

También se han realizado dos experimentos adicionales con instancias nuevas del problema uno modificando la ubicación inicial de las personas y otra modificando tanto el estado inicial y

los objetivos. Los resultados de ambas pruebas se muestran en las tablas Tabla 2.2 y Tabla 2.3 respectivamente.

Planificador	Acciones	Duracion	tiempo
LPG	30	**128**	21.23
OPTIC	34	149	6.40

Tabla 2.2. Resultados de los planificadores LPG y OPTIC variando el estado inicial de las personas.

Planificador	Acciones	Duracion	tiempo
LPG	25	**92**	10
OPTIC	28	114	0.30

Tabla 2.3. Resultados de los planificadores LPG y OPTIC variando el estado inicial y objetivo.

Como ya se mencionó, los planes del planificador LPG se realizaron con 5 iteraciones a diferencia de 3 y se observa que le toma más tiempo en encontrar un plan pero resuelve mejor el problema.

3. Dominio con recursos numéricos

En este apartado se considera el problema añadiendo un recurso que sea inversamente proporcional al tiempo.

3.1. Consideración de la energía como recurso

Hemos elegido como recurso la energía de cada ascensor. Este recurso toma la forma de una batería por cada ascensor. Cada acción cuesta una cantidad de energía fija para el get-on y get-down y una cantidad de energía variable para el move-up y move-down. La fórmula utilizada para definir la cantidad de energía que cuesta la acción depende del número de pisos que debe pasar, de la velocidad del ascensor y del ratio de descarga de la batería.

3.2. Modificaciones sobre el dominio temporal

Para poder modelar la energía, hemos añadido tres funciones:

- **(uncharge ?l - lift):** que indica el ratio de descarga del ascensor
- **(battery-level ?l - lift):** que indica el nivel de batería del ascensor
- **(total-energy-used):** que indica el valor total de energía que ha sido utilizada

En cuanto a las acciones, hemos añadido precondiciones para que un ascensor solo pueda moverse entre dos pisos si tiene energía suficiente. También añadimos otros efectos a estas acciones. Primero estamos disminuyendo el nivel de batería del ascensor y en segundo lugar estamos incrementando el total de energía que hemos utilizado. Podemos ver en la figura 3.1. Un ejemplo para el operador move-up-slow.

```
(:durative-action move-up-slow
      :parameters (?l - lift-slow ?f1 - integer ?f2 - integer)
      :duration (= ?duration (travel-slow ?f1 ?f2))
      :condition (and (at start(lift-at ?l ?f1)) (at start(can-go ?l ?f2)) (at start(>(floor ?f2) (floor ?f1) ))
      (at start (>= (battery-level ?l) (* (- (floor ?f2)(floor ?f1))(* (uncharge ?l) (travel-slow ?f1 ?f2))))))
      :effect
      (and (at start (not (lift-at ?l ?f1))) (at end  (lift-at ?l ?f2)) (at end  (lift-at ?l ?f2))
       (at end (decrease (battery-level ?l) (* (- (floor ?f2)(floor ?f1))(* (uncharge ?l) (travel-slow ?f1 ?f2)))))
       (at end (increase (total-energy-used) (* (- (floor ?f2)(floor ?f1))(* (uncharge ?l) (travel-slow ?f1 ?f2)))))
       ))
```

Figura 3.1. Implementación del operador *move-up-slow*

3.3. Recursos renovables

En el caso del dominio donde el recurso es renovable hemos añadido 2 funciones:

- **(have-charger ?floor):** que indica si hay un cargador o no en el piso

- **(battery-capacity ?lift):** que indica la capacidad de la batería del ascensor

y una acción recharge que permite recargar los ascensores. La podemos ver en la figura XXX.

```
(:durative-action recharge
    :parameters (?l - lift ?f - integer)
    :duration (= ?duration (/ (- (battery-capacity ?l) (battery-level ?l)) (charge ?l)))
    :condition (and
        (at start (and
            (have-charger ?f)))
        (over all (lift-at ?l ?f))
    )
    :effect ( at end (assign (battery-level ?l) (battery-capacity ?l)))
)
```

Figura 3.2. Implementación del operador *recharge*

3.4. Cambios en el problema

En el problema estamos iniciando, el nivel de batería (battery-level) de cada ascensor, el ratio de descarga (uncharge) de cada ascensor y la energía total (total-energy-used). En el caso de recursos renovables también estamos iniciando la capacidad de batería (battery-level) de cada ascensor, el ratio de carga (charge) de cada ascensor y donde se encuentran los cargadores (can-charge). Podemos ver la implementación en la figura 3.3.

```
;============================ nivel de bateria asencores ==============
(= (battery-level lift1) 300) (= (battery-level lift2) 300) (= (battery-level lift3) 300) (= (battery-level lift4) 300)
(= (battery-level lift5) 300)

;============================ capacidad bateria asencores ==============
(= (battery-capacity lift1) 300) (= (battery-capacity lift2) 300) (= (battery-capacity lift3) 300)
(= (battery-capacity lift4) 300) (= (battery-capacity lift5) 300)

;============================ uncharge valores ==============
(= (uncharge lift1) 2 ) (= (uncharge lift2) 2 ) (= (uncharge lift3) 2 ) (= (uncharge lift4) 2 ) (= (uncharge lift5) 2 )

;============================ charge valores ==============
(= (charge lift1) 8 ) (= (charge lift2) 8 ) (= (charge lift3) 8 ) (= (charge lift4) 8 ) (= (charge lift5) 8 )
```

```
;==================== donde estan los cargadores ==========================
(have-charger n4) (have-charger n8) (have-charger n6) (have-charger n2) (have-charger n10)

(= (total-energy-used) 0)
```

Figura 3.3. Cambios hechos en el problema

3.5. Experimentación y evaluación

Hemos probado esto con LPG y Optic, solo LPG nos daba una solución. Pensamos que se debe a la expresión que nos da la energía que se consume que no es lineal.

Podemos ver en la tabla 3.1. los resultados con LPG cuando los recursos no son renovables,

minimizando el recurso y después el tiempo. Es normal que la duración sea menor en el caso de minimizar el tiempo que en el caso de minimizar el recurso.

	Minimizando el recurso	Minimizando el tiempo
N° Pasos	41	37
Duracion	168	140
Tiempo total	11.66	14
Recurso	138.40	X

Tabla 3.1. Resultados con LPG en el caso de recursos no renovables

Podemos ver en la tabla 3.2. los resultados con LPG cuando los recursos son renovables, minimizando el recurso y después el tiempo. Vemos, como antes, que la duración es menor minimizando el tiempo.

	Minimizando el recurso	Minimizando el tiempo
N° Pasos	58	43
Duracion	250	172
Tiempo total	64	102.02
Recurso	764.00	X

Tabla 3.2. Resultados con LPG en el caso de recursos renovables

4. Desarrollo parcial de un árbol POP

En esta sección se desarrolla un plan de orden parcial. Se ha escogido un solo objetivo a partir del problema base que se ha visto en el dominio proposicional. En este caso, se ha elegido el objetivo **(passenger-at p2 n12)**. Para la selección del flaw en cada nodo del árbol, se ha optado por una elección aleatoria. A continuación, se detalla el desarrollo del árbol.
Para simplificar, hemos eliminado las precondiciones relativas a los lugares donde pueden ir los ascensores. (can-go ?lift ?floor) y las precondiciones que nos indican donde se encuentra un piso relativo al otro (above ?floor ?floor).

Figura 4.1. Representación del nodo inicial del árbol POP, el plan 1

A partir de este primer plan, se decide resolver el literal del objetivo final, que se resuelve con la acción de get-down, la cual descarga a la persona en el piso 12. En la figura 4.2 podemos ver el estado del árbol al que se llega.

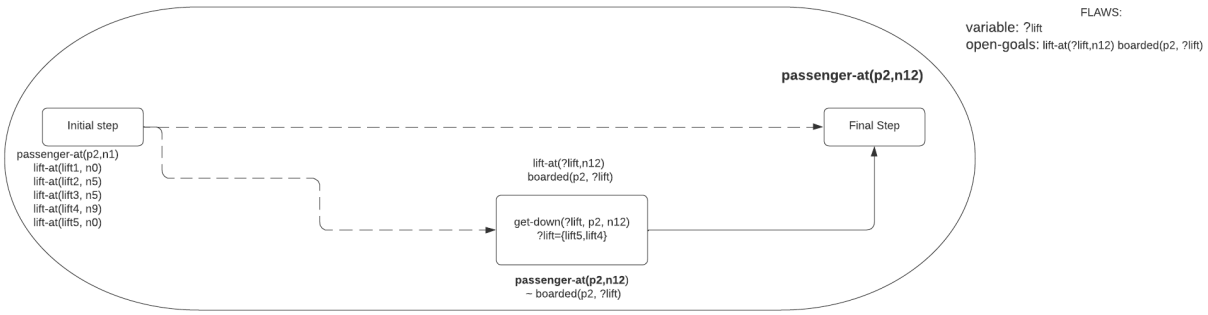


Figura 4.2. Representación del plan 2

Una vez estamos en el plan 2 del árbol POP, tenemos tres flaws: los literales (lift-at ?lift n12) y (boarded p2 ?lift), y la instanciación de la variable ?lift. En este caso se ha decidido resolver la variable ?lift y elegimos el ascensor 5. De este modo, se llega al siguiente nodo del árbol, que podemos observar en la figura 4.3.

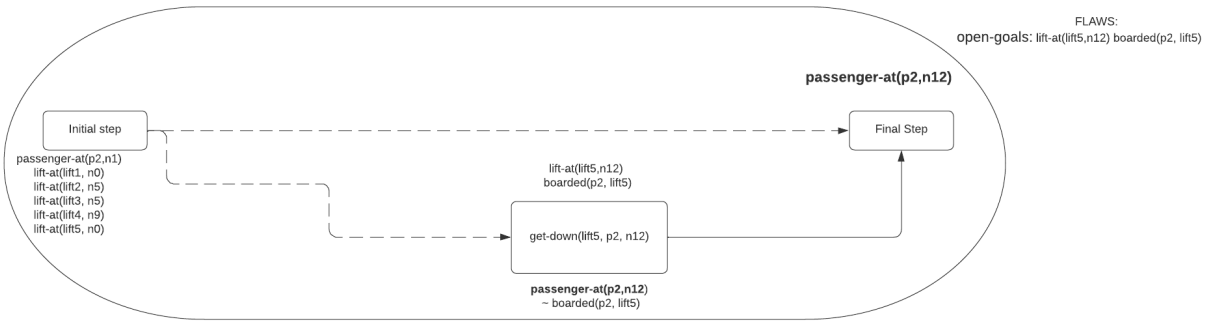


Figura 4.3. Representación del plan 2.1

Una vez estamos en el plan 2.1 del árbol POP, tenemos dos flaws que son los literales (lift-at ?lift n12) y (boarded p2 ?lift). En este caso se ha decidido resolver el literal (lift-at ?lift n12). Este literal se resuelve con acciones de mover el ascensor 5 de un piso ?floor hasta el piso n12. Se llega al siguiente nodo que podemos ver en la figura XXXX.

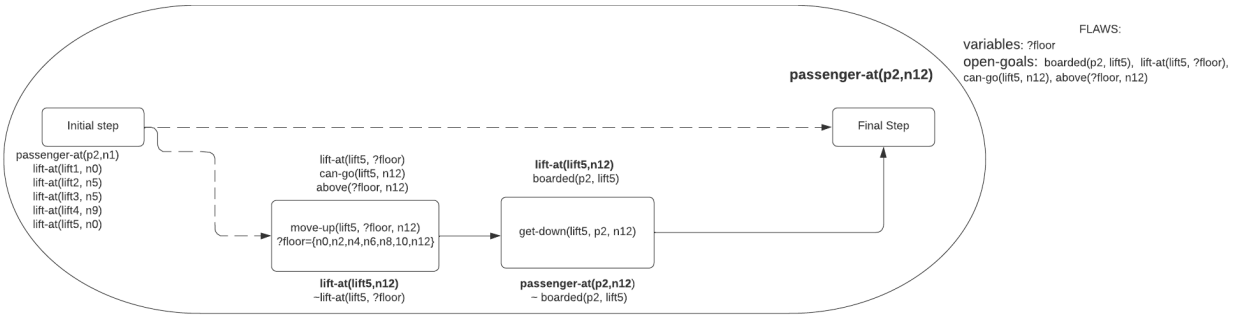
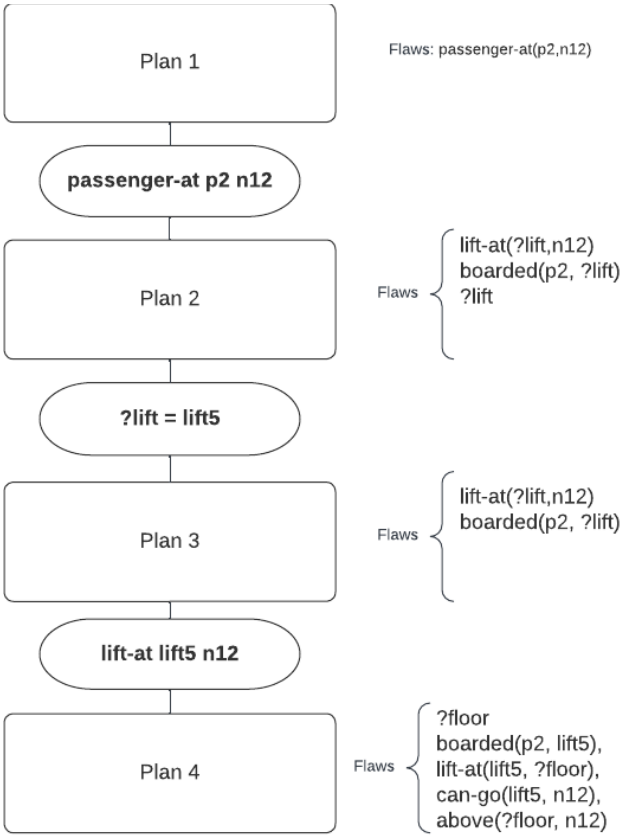


Figura 4.3. Representación del plan 3

En la siguiente figura podemos ver un resumen de los nodos del árbol POP que hemos generado. Se puede observar que no ramifica en ningún momento. Esto ocurre porque no hay más de una forma de resolver los flaws que se han elegido.


Plan 1
Flaws: passenger-at(p2,n12)
passenger-at p2 n12
Plan 2
Flaws
lift-at(?lift,n12)
boarded(p2, ?lift)
?lift
?lift = lift5
Plan 3
Flaws
lift-at(?lift,n12)
boarded(p2, ?lift)
lift-at lift5 n12
Plan 4
Flaws
?floor
boarded(p2, lift5),
lift-at(lift5, ?floor),
can-go(lift5, n12),
above(?floor, n12)

5. Graphplan

En esta sección se desarrolla un grafo de planificación relajado. Este grafo sirve para calcular la heurısticas que den información al planificador para que pueda dirigir la búsqueda. Estas heurısticas son fáciles de extraer ya que es un grafo multietapa donde el número de capas hasta llegar al objetivo es un límite inferior del número de acciones del plan. Se ha calculado un grafo de planificación para resolver el problema mencionado con los objetivos pasajero 0 a la planta 3 y pasajero 1 a la planta 11 considerando el dominio proposicional. El grafo se puede encontrar adjunto en un fichero .xlsx en la entrega.

5.1. Heuristicas h_sum y h_max

$$h\_max(G) = max(4,7) = 7$$

$$h\_max(G) = 4 + 7 = 11$$

5.2. Plan Relajado

La extracción del plan relajado, tal y como se ha mencionado anteriormente, se realiza en el orden inverso a la construcción del grafo de planificación relajado. Partiremos desde el último nivel, observaremos qué objetivos se han alcanzado en dicho nivel y apuntaremos las acciones que han permitido alcanzarlos (destacadas en rojo en el fichero .xlsx).

Nivel 1:
(move-up lift1 n0 n2)
(move-down lift3 n5 n4)
(move-down lift4 n9 n8)

Nivel 2:
(get-on lift1 p0 n2)
(get-on lift3 p1 n4)

Nivel 3:
(move-up lift1 n2 n3)
(move-up lift3 n4 n8)

Nivel 4:
(get-down lift1 p0 n3)
(get-down lift3 p1 n8)

Nivel 5:
(get-on lift4 p1 n8)

Nivel 6:
(move-up lift4 n8 n11)

Nivel 7:

(get-down lift4 p1 n11)

Este plan se puede extraer en tiempo polinómico y sin necesidad de hacer backtracking ya que el grafo de planificación que hemos calculado es relajado, por lo que no tiene efectos negativos y no hay que calcular mutex. Una vez se consigue un literal por primera vez se cumple para siempre, y la extracción del plan es directa.

$$plan\_relajado(G) = 12$$

5.3. Heurística mas informada

Cogemos la heurística ***h_sum*** como la más informada, la ***h_max*** solo tendría en cuenta el nivel donde se consigue el último objetivo, proporcionando menos información respecto al coste restante para llegar a la solución. Por otro lado, la heurística plan relajado (con un valor de 8 acciones) también tendría un alto grado de información, pero dado que muchas de las acciones se ejecutan simultáneamente en el plan relajado (sucede del mismo modo con la consecución de objetivos pero en menor medida), su estimación sería más pobre que la ***h_sum***.

Conclusiones

Primero se llevó a cabo una codificación en PDDL del problema. La codificación del dominio proposicional no fue difícil debido a una ligera experiencia previa con este lenguaje.
En la parte experimental, se aplicaron los conceptos aprendidos en clases para entender el funcionamiento de los diferentes planificadores. La parte experimental fue en donde se aprendió más.
Los ejercicios de árbol de estados POP y Graphplan ayudaron a aplicar los conceptos explicados en clases y a entender mejor el mundo de los planificadores, ya que proporcionaron una imagen de cómo funcionan internamente al momento de planificar.
En general, el proceso de desarrollo y experimentación nos brindó una buena base sobre el tema de planificación inteligente, y nos proporcionó conocimiento y experiencia para comprender cómo abordar tareas de planificación inteligente.