Planificación del puerto

Marcos Esteve Casademunt, David Gimeno Gómez

Febrero 2020

Contents

1 Introducción al dominio del puerto

El dominio proposicional del puerto consiste en un conjunto de muelles de descarga donde cada uno de ellos contienen varios contenedores. Además, en un momento dado, la compañía de transporte solicita que varios de los contenedores, a partir de ahora objetivos, deberán estar en el muelle 1 para su recogida y transporte.

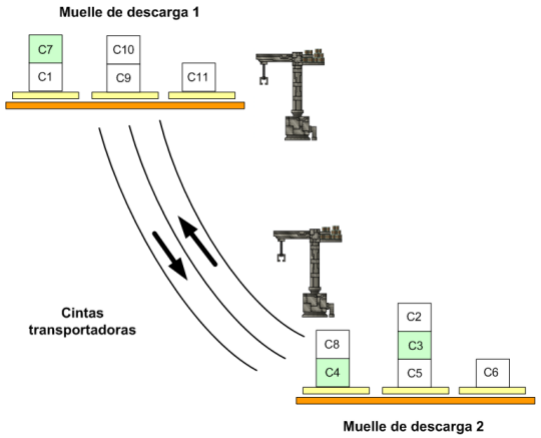

Figure 1: Dominio propuesto en el enunciado del problema

Tal y como se puede ver en la figura 1 cada muelle de descarga estará compuesto por distintas pilas. Estas pilas, a su vez, contendrán un conjunto de contenedores objetivos (verdes) o no objetivos (blancos). Además, cada muelle dispondrá de una grúa que se encargará de cargar y descargar contenedores de las distintas pilas o bien ponerlos en una cinta transportadora para que sean enviados a otro muelle de descarga.

Por último, el objetivo final consiste en que aquellos contenedores que sean objetivos deberán de estar en el muelle de descarga 1 y además todos ellos deberán de estar disponibles para su posterior recogida. Esto es, o bien deberán de ser la cima de la pila (no tener un contenedor no objetivo por encima) o en caso de tener un contenedor encima este deberá de ser también objetivo. Un posible ejemplo de lo descrito se puede apreciar en la figura 2.

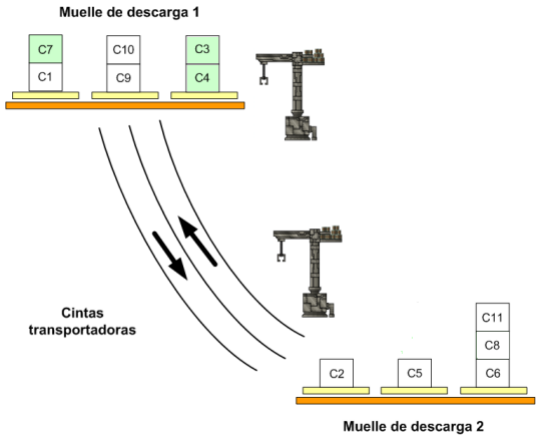

Figure 2: Posible solución al dominio propuesto en el enunciado del problema

2 Dominio proposicional

A continuación pasamos a comentar el diseño de los distintos predicados así como las distintas acciones que se podrán aplicar en nuestro dominio.

2.1 Diseño de los predicados

- **en ?c** - (either contenedor pila grua) **?p** - (either pila muelle cinta grua)

  El predicado **en** trata de modelar de una forma genérica la relación existente entre los distintos objetos. De esta forma conseguimos modelar cuando un contenedor se encuentra en una grúa o en una cinta, una grúa está en un muelle etc.

- **encima ?c1** - contenedor **?c2** - (either contenedor pila)

  Por otra parte, el predicado **encima** permite modelar cuando un contenedor c1 se encuentra encima de un contenedor o una pila (en caso de no tener un contenedor por debajo). De esta forma conseguimos modelar el comportamiento de apilar contenedores uno encima de otro.

- **top ?c** - (either contenedor pila) **?p** - pila

  El predicado **top** permite modelar cuando un contenedor es la cima de una pila. Además destacar que con este predicado podemos modelar que la cima de una pila puede ser la pila misma, esto nos permite identificar cuando una pila está vacía.

- **libre ?x** - (either grua cinta)

El predicado **libre** nos permite modelar cuando una grúa o una cinta está libre para poder transportar paquetes.

- **conecta ?c** - cinta **?m1** - muelle **?m2** - muelle

  Mediante el predicado **conecta** se consigue modelar cuando una cinta **c** está conectando dos muelles **m1** y **m2**

- **next ?n1** - nivel **?n2** - nivel y **altura ?p** - pila **?n** - nivel

  El predicado **next** por su parte, permite modelar cuando existe un nivel posterior o inferior en la pila, de esta forma, podemos diseñar mediante predicados lógicos la existencia de un nivel máximo de pila. El predicado next opera junto al predicado **altura** el cual nos permite consultar cuál es la **altura** de una **pila**.

- **disponible ?c** - (either contenedor pila)

  El predicado **disponible** permite modelar cuando un contenedor está disponible para ser recogido. El comportamiento de este predicado es complejo y se detallará en la regla dejar.

- **verde ?c** - contenedor

  Por último, el predicado verde nos permite modelar cuando un contenedor es objetivo

2.2 Diseño de los operadores

Una vez tengamos definidos los distintos predicados que nos permiten expresar las diferentes situaciones posibles de nuestro dominio, es hora de construir los operadores con los que modelaremos las acciones que podemos realizar. En otras palabras, estos operadores provocarán cambios sobre el estado actual del puerto, siempre y cuando se cumplan una serie de condiciones, con la finalidad de alcanzar los objetivos propuestos. Más concretamente, seremos capaces de coger y dejar contenedores con la grúa, así como recogerlos y colocarlos en la cinta transportadora para llevarlos a otros muelles. De esta manera, al final logramos modelar todo el dominio proposicional con seis operadores. Para comprender mejor cómo es su modelado, la definición y construcción de estos operadores se muestra en los sucesivos subapartados.

2.2.1 Coger un contenedor con la grúa

El primero de los operadores se encargará de la acción de coger con la grúa un contenedor de una pila para su posterior colocación en la misma u otra pila o, incluso, su traslado a la cinta transportadora; pero esto ya son consecuencias que serán controladas por otros operadores. En lo que respecta al

operador coger, es necesario tener en cuenta que sólo podrán cogerse aquellos contenedores que se encuentren en la cima de alguna pila y, además, que esta pila se encuentre en el mismo muelle que la grúa. Una vez cogido el contenedor, debemos actualizar la altura de la pila, tal y como se comentó en anteriores apartados. Además, un nuevo contenedor pasa a ser la nueva cima de la pila y, en consecuencia, a establecerse como un contenedor disponible independientemente del tipo que sea (objetivo o no). El resto de precondiciones y efectos son bastantes razonables y pueden observarse en la codificación mostrada en la Figura 3. Al final, logramos modelar esta acción con la definición de un único operador.

```
(:action coger
    :parameters (?g - grua ?p - pila ?c1 - contenedor ?c2 -(either contenedor pila) ?m - muelle  ?nOrigen - nivel ?nDestino  - nivel )
    :precondition (and
                    (en ?p ?m) ; pila estar en muelle
                    (en ?g ?m ) ; grua estar en muelle
                    (libre ?g)  ; grua estar libre
                    (top ?c1 ?p) ; contenedor estar en la cima de la pila
                    (encima ?c1 ?c2) ; contenedor estar encima de c2 siendo este un contenedor, contenedorObjetivo o una pila
                    (en ?c1 ?p); contenedor estar en la pila
                    (altura ?p ?nOrigen) ;la altura de la pila es norigen
                    (next ?nDestino ?nOrigen) ; existe una altura menor
                    (disponible ?c1)

                )
    :effect (and
            (en ?c1 ?g) ; el contenedor esta en la grua
            (disponible ?c2) ; el contenedor c2 pasa a estar disponible ya que es el nuevo top
            (altura ?p ?nDestino) ; la altura se decrementa
            (en ?c2 ?p)
            (top ?c2 ?p); cambiamos el nuevo top
            (not (en ?c1 ?p)) ; el contenedor ya no esta en la pila
            (not (top ?c1 ?p)); no esta en el top
            (not (encima ?c1 ?c2)) ;el contenedor c1 ya no esta de c2
            (not (libre ?g)) ;La grua ya no está libre
            (not (altura ?p ?nOrigen)) ; altura incorrecta eliminada
            (not (disponible ?c1)) ; el contenedor en la grua no esta disponible

    )
)
```

Figure 3: Regla coger

2.2.2 Dejar un contenedor con la grúa

Respecto a la acción de dejar un contenedor pueden suceder cuatro escenarios, tal y como podemos observar en la Figura 4. Siempre y cuando consideremos que dejar un contenedor sobre una pila vacía es como dejarlo sobre un contenedor no objetivo.

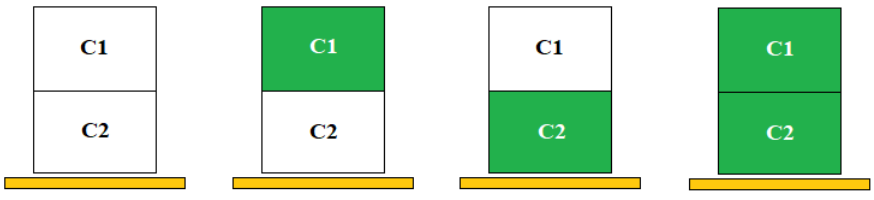

Figure 4: Esquema de posibilidades en la regla dejar

Las tres primeras situaciones provocarían el mismo efecto, y es que el contenedor C2 ya no estaría disponible; mientras que en el último de los casos, cuando ambos contenedores son objetivos, esta disponibilidad del contenedor C2 se conservaría. Por ello, en una primera aproximación pensamos en definir dos operadores que modelasen la acción. Un operador que se encargarse de las tres primeras situaciones, ya que comparten el mismo efecto; y otro operador centrado en el último de los escenarios. Sin embargo, surgía un problema debido a la gran flexibilidad que presenta el primero de los operadores definidos, puesto que éste permite que C1 y C2 puedan ser o no contenedores objetivos. Por lo tanto, cuando se diese la situación del último caso, se instanciarían ambos operadores pudiendo provocar efectos no deseados. Por estas razones, al final hemos optado por definir tres operadores que abordan toda posibilidad de situaciones sin interferir con otros operadores, tal y como se muestra en la Figura 5. De este modo, tendríamos un operador encargado de dejar un contenedor cualquiera sobre otro que no sea objetivo, un segundo operador para dejar un contenedor no objetivo sobre uno que sí lo es y un último operador que se encargue del último de los casos descritos.

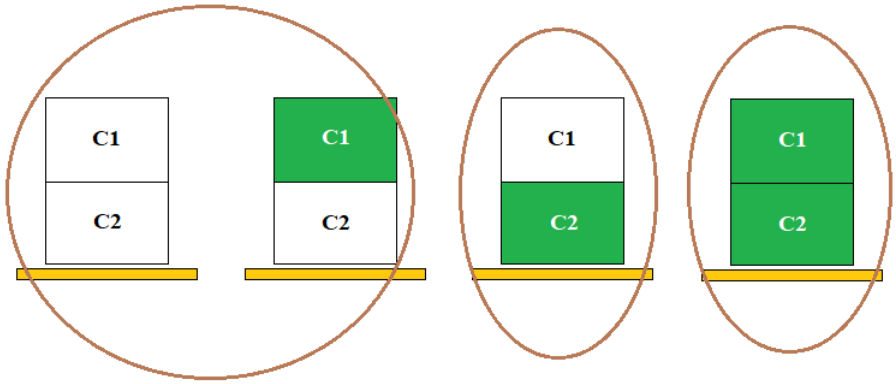

Figure 5: Esquema unificado de posibilidades en la regla dejar

Con todo esto, ya podemos visualizar la codificación de los operadores mencionados en PDDL mediante los predicados comentados en anteriores apartados de la memoria. Como se puede observar la implementación de los tres operadores es muy similar, simplemente destacar los pequeños detalles que se han ido explicando en esta sección. Estos detalles han sido remarcados sobre el código mostrado en las siguientes Figuras 6, 7 y 8 para facilitar su comparación.

```
(:action dejarSobreNoVerde
    :parameters (?c1 -contenedor ?c2 - (either contenedor pila) ?g - grua ?m - muelle ?p - pila ?nOrigen -nivel ?nDestino  -nivel)
    :precondition (and
        (en ?c1 ?g) ; el contenedor debe estar en la grua
        (en ?g ?m) ; la grua debe estar en el muelle origen
        (en ?p ?m); la pila debe estar en el muelle
        (en ?c2 ?p); c2 debe estar en la pila
        (top ?c2 ?p); el contenedor 2 debe ser la cima de la pila
        (noverde ?c2) ; el contenedor en la pila no debe ser verde
        (altura ?p ?nOrigen) ;la altura de la pila es norigen
        (next ?nOrigen ?nDestino) ; existe una altura menor
    )
    :effect (and
        (libre ?g) ;la grua pasa a estar libre
        (en ?c1 ?p) ; el contenedor esta en la la cinta
        (encima ?c1 ?c2) ; c1 esta encima de c2
        (top ?c1 ?p) ; c1 pasa a ser el top de la pila p
        (disponible ?c1) ; c1 pasa a estar disponible
        (altura ?p ?nDestino) ; la altura se decrementa
        (not (en ?c1 ?g)) ; el contenedor no esta en la grua
        (not (top ?c2 ?p)) ;c2 ya no es la cima de la pila
        (not (disponible ?c2)) ; c2 ya no esta disponible
        (not (altura ?p ?nOrigen)) ; altura incorrecta eliminada

    )
)
```

Figure 6: Regla dejar sobre no verde

```
(:action dejarBlancoSobreVerde
    :parameters (?c1 - contenedor ?c2 - contenedor ?g - grua ?m - muelle ?p - pila ?nOrigen - nivel ?nDestino  - nivel)
    :precondition (and
        (en ?c1 ?g) ; el contenedor debe estar en la grua
        (en ?g ?m) ; la grua debe estar en el muelle origen
        (en ?p ?m); la pila debe estar en el muelle
        (en ?c2 ?p); c2 debe estar en la pila
        (top ?c2 ?p); el contenedor 2 debe ser la cima de la pila
        (verde ?c2) ; el contenedor en la pila no debe ser verde
        (altura ?p ?nOrigen) ;la altura de la pila es norigen
        (next ?nOrigen ?nDestino) ; existe una altura menor
        (noverde ?c1) ; c1 no debe ser verde
    )
    :effect (and
        (libre ?g) ;la grua pasa a estar libre
        (en ?c1 ?p) ; el contenedor esta en la la cinta
        (encima ?c1 ?c2) ; c1 esta encima de c2
        (top ?c1 ?p) ; c1 pasa a ser el top de la pila p
        (disponible ?c1) ; c1 pasa a estar disponible
        (altura ?p ?nDestino) ; la altura se decrementa
        (not (en ?c1 ?g)) ; el contenedor no esta en la grua
        (not (top ?c2 ?p)) ;c2 ya no es la cima de la pila
        (not (disponible ?c2)) ; c2 ya no esta disponible
        (not (altura ?p ?nOrigen)) ; altura incorrecta eliminada

    )
)
```

Figure 7: Dejar blanco sobre verde

```
(:action dejarVerdeSobreVerde
    :parameters (?c1 -contenedor ?c2 - contenedor ?g - grua ?m - muelle ?p - pila ?nOrigen -nivel ?nDestino  -nivel)
    :precondition (and
        (en ?c1 ?g) ; el contenedor debe estar en la grua
        (en ?g ?m) ; la grua debe estar en el muelle origen
        (en ?p ?m); la pila debe estar en el muelle
        (en ?c2 ?p); c2 debe estar en la pila
        (top ?c2 ?p); el contenedor 2 debe ser la cima de la pila
        (verde ?c2) ; el contenedor en la pila  debe ser verde
        (altura ?p ?nOrigen) ;la altura de la pila es norigen
        (next ?nOrigen ?nDestino) ; existe una altura menor
        (verde ?c1) ;c1 debe ser verde
    )
    :effect (and
        (libre ?g) ;la grua pasa a estar libre
        (en ?c1 ?p) ; el contenedor esta en la la cinta
        (encima ?c1 ?c2) ; c1 esta encima de c2
        (top ?c1 ?p) ; c1 pasa a ser el top de la pila p
        (disponible ?c1) ; c1 pasa a estar disponible
        (altura ?p ?nDestino) ; la altura se decrementa
        (not (en ?c1 ?g)) ; el contenedor no esta en la grua
        (not (top ?c2 ?p)) ;c2 ya no es la cima de la pila
        (not (altura ?p ?nOrigen)) ; altura incorrecta eliminada

    )
)
```

Figure 8: Dejar verde sobre verde

2.2.3 Coger un contenedor de la cinta

Por otra parte, debemos ser capaces de recoger un contenedor que nos haya llegado desde otro muelle a través de la cinta transportadores correspondiente. En este operador necesitaremos conocer la grúa, el contenedor, la cinta y, claro está, los dos muelles involucrados en la transacción. Su codificación en PDDL puede observarse en la Figura 9. Lo único a destacar es que tenemos que definir adecuadamente de que la grúa que empleemos sea la del muelle destinatario.

```
(:action cogerDeCinta
    :parameters (?c - contenedor ?g - grua ?mOrigen - muelle ?mDestino - muelle ?t - cinta)
    :precondition (and
        (en ?c ?t) ; el contenedor debe estar en la cinta
        ;(not(libre ?t)) ;cinta no libre
        ;(nolibre ?t)
        (en ?g ?mDestino) ; la grua debe estar en el muelle destino
        (conecta ?t ?mOrigen ?mDestino ); debe existir una conexion entre el muelle origen y el muelle destino
        (libre ?g) ; la grua debe estar libre
    )
    :effect (and
        (not(libre ?g)) ;la grua ya no estar libre
        ;(nolibre ?g)
        (en ?c ?g) ; el contenedor esta en la la grua
        (not (en ?c ?t)) ; el contenedor no esta en la cinta
        (libre ?t) ; la cinta esta ocupada
        (en ?c ?mDestino) ;el contenedor ya tiene ubicacion
        ;(not (nolibre ?t)) ; PUEDE SER ESTO??
    )
)
```

Figure 9: Coger de cinta

2.2.4 Dejar un contenedor en la cinta

De igual modo, hemos de modelar la acción de dejar un contenedor en la cinta para su posterior transporte hacia un muelle destino. No deja de ser el caso opuesto al planteamiento anterior por lo que tendrá una codificación similar. Podemos ver su implementación en la Figura 10.

```
(:action dejarEnCinta
    :parameters (?c -contenedor ?g - grua ?mOrigen - muelle ?mDestino - muelle ?t - cinta)
    :precondition (and
        (en ?c ?g) ; el contenedor debe estar en la grua
        (en ?g ?mOrigen) ; la grua debe estar en el muelle origen
        (conecta ?t ?mOrigen ?mDestino ); debe existir una conexion entre el muelle origen y el muelle destino
        (libre ?t) ; la cinta debe estar libre
        (en ?c ?mOrigen) ;el contenedor esta en el muelle origen
     )
    :effect (and
        (libre ?g) ;la grua pasa a estar libre
        (en ?c ?t) ; el contenedor esta en la la cinta
        (not (en ?c ?g)) ; el contenedor no esta en la grua
        (not (libre ?t)) ; la cinta esta ocupada
        (not (en ?c ?mOrigen)) ;esta en la frontera "andorra"
    )
)
```

Figure 10: Dejar en cinta

2.3 Definición del problema

Para la definición del estado inicial del problema para la instancia descrita en la figura 1. Será necesario definir todos los predicados necesarios para la inicialización del problema, estos predicados deberán indicar:

- Cada contenedor en su respectiva pila → Predicado **en**
- Cada pila en su respectivo muelle → Predicado **en**
- Cada grúa en su respectivo muelle → Predicado **en**
- Cada contenedor encima de que contenedor o pila está → Predicado **encima**
- Las conexiones entre muelles mediante cintas → Predicado **conecta**
- Las cimas de cada pila → Predicado **top**
- Las alturas de cada pila → Predicados **next** y **altura**
- Los contenedores que están disponibles → Predicado **disponible**
- Los contenedor que son objetivo → Predicado **verde**

Además podemos codificar en PDDL el objetivo final del problema como:

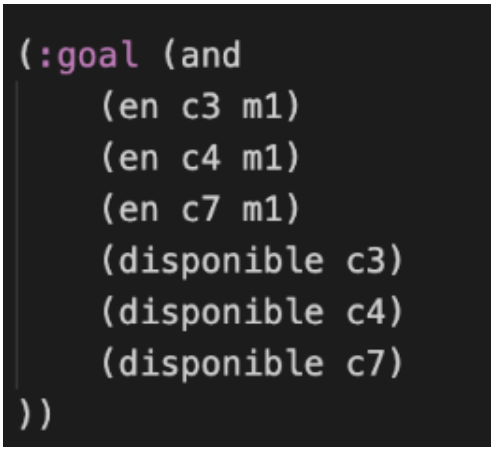

Figure 11: Objetivo codificado en PDDL

Donde todos los contenedores que son objetivos deben de estar en el muelle 1 y además deben de estar disponibles para su posterior carga.

2.4 Experimentación y resultados

El objetivo principal de este punto es realizar una evaluación variando los distintos parámetros del dominio inicial. Para ello, en primer lugar, realizaremos una evaluación con el problema propuesto en el boletín que puede observarse en la Figura 1. Tras esto, evaluaremos los distintos planificadores con distintos tamaños de problema. Por último, estudiaremos cuáles pueden ser los posibles cuellos de botella a la hora de resolver un determinado problema.

```
 0: COGER G2 P5 C2 C3 M2 N3 N2
 1: DEJARSOBRENOVERDE G2 P6 C2 C6 M2 N1 N2
 2: COGER G2 P4 C8 C4 M2 N2 N1
 3: DEJARSOBRENOVERDE G2 P6 C8 C2 M2 N2 N3
 4: COGER G2 P5 C3 C5 M2 N2 N1
 5: DEJARENCINTA G2 C3 M2 M1 T2
 6: COGERDECINTA G1 C3 M2 M1 T2
 7: DEJARVERDESOBREVERDE G1 P1 C3 C7 M1 N2 N3
 8: COGER G2 P4 C4 P4 M2 N1 N0
 9: DEJARENCINTA G2 C4 M2 M1 T2
10: COGERDECINTA G1 C4 M2 M1 T2
11: DEJARSOBRENOVERDE G1 P3 C4 C11 M1 N1 N2
```

Figure 12: Plan obtenido por Metric FF para el enunciado del problema

En la figura 12 podemos observar el plan devuelto por el planificador Metric FF al problema propuesto en el enunciado. Como sabemos, Metric

FF es un planificador secuencial, por lo que los planes devueltos no incluyen paralelismo.

```
0.0000: (COGER G2 P5 C2 C3 M2 N3 N2) [D:1.00; C:1.00]
1.0000: (DEJARSOBRENOVERDE G2 P6 C2 C6 M2 N1 N2) [D:1.00; C:1.00]
2.0000: (COGER G2 P5 C3 C5 M2 N2 N1) [D:1.00; C:1.00]
3.0000: (DEJARENCINTA G2 C3 M2 M1 T2) [D:1.00; C:1.00]
4.0000: (COGERDECINTA G1 C3 M2 M1 T2) [D:1.00; C:1.00]
4.0000: (COGER G2 P4 C8 C4 M2 N2 N1) [D:1.00; C:1.00]
5.0000: (DEJARSOBRENOVERDE G2 P6 C8 C2 M2 N2 N3) [D:1.00; C:1.00]
5.0000: (DEJARVERDESOBREVERDE G1 P1 C3 C7 M1 N2 N3) [D:1.00; C:1.00]
6.0000: (COGER G2 P4 C4 P4 M2 N1 N0) [D:1.00; C:1.00]
7.0000: (DEJARENCINTA G2 C4 M2 M1 T2) [D:1.00; C:1.00]
8.0000: (COGERDECINTA G1 C4 M2 M1 T2) [D:1.00; C:1.00]
9.0000: (DEJARSOBRENOVERDE G1 P2 C4 C10 M1 N2 N3) [D:1.00; C:1.00]
```

Figure 13: Plan obtenido por LPG para el enunciado del problema

Respecto al planificador LPG, podemos observar en la figura 13 como el plan obtenido por el planificador es paralelo; por ejemplo, en el instante temporal 4 la grúa 1 está cogiendo un contenedor de la cinta en el muelle 1 mientras que la grúa 2 está cogiendo un contenedor en el muelle dos. También se podría observar un comportamiento similar en el instante temporal 5.

```
0.000: (coger g2 p4 c8 c4 m2 n2 n1)  [0.001]
0.001: (dejarencinta g2 c8 m2 m1 t2)  [0.001]
0.002: (coger g2 p5 c2 c3 m2 n3 n2)  [0.001]
0.002: (cogerdecinta g1 c8 m2 m1 t2)  [0.001]
0.003: (dejarsobrenoverde g2 p6 c2 c6 m2 n1 n2)  [0.001]
0.003: (dejarsobrenoverde g1 p3 c8 c11 m1 n1 n2)  [0.001]
0.004: (coger g2 p4 c4 p4 m2 n1 n0)  [0.001]
0.005: (dejarencinta g2 c4 m2 m1 t2)  [0.001]
0.006: (cogerdecinta g1 c4 m2 m1 t2)  [0.001]
0.006: (coger g2 p5 c3 c5 m2 n2 n1)  [0.001]
0.007: (dejarsobrenoverde g1 p2 c4 c10 m1 n2 n3)  [0.001]
0.007: (dejarencinta g2 c3 m2 m1 t2)  [0.001]
0.008: (cogerdecinta g1 c3 m2 m1 t2)  [0.001]
0.009: (dejarsobrenoverde g1 p3 c3 c8 m1 n2 n3)  [0.001]
```

Figure 14: Plan obtenido por Optic para el enunciado del problema

Por último, respecto al planificador Optic, podemos observar en la figura 14 un comportamiento similar al obtenido por el planificador LPG. Podemos destacar que existe un comportamiento paralelo en los instantes temporales 2, 3, 6 y 7.

En cuanto a la evaluación de los planificadores respecto al tamaño del problema, hemos definido los puertos expuestos en el Tabla 1 donde indicamos todos los detalles referentes a su tamaño como puedan ser el número de contenedores, muelles, entre otros. Cabe destacar que el problema de menor tamaño coincide con el problema del boletín y del que acabamos de comentar los planes obtenidos.

Table 1: Tamaños propuestos para la evaluación temporal del problema

Tamaño	**Muelles**	**Contenedores**	**Objetivos**	**Pilas**	**Max pila**
Pequeño	2	11	3	6	3
Mediano	3	16	5	9	3
Grande	4	20	7	12	4

Una vez hayamos realizado todas las pruebas con cada uno de los planificadores, hemos optado por evaluarlos respecto al tiempo de resolución. Para facilitar esta comparación se dispone de la tabla 2. En ella podemos observar como con el problema pequeño todos los planificadores obtienen el plan en un tiempo reducido, siendo LPG el peor de ellos. Por otro lado, al aumentar la talla del problema a su tamaño mediano vemos como LPG sufre un aumento notable alcanzando cerca de 54 segundos. Este tiempo es claramente superior al resto de sus competidores, entre los cuales optaríamos, tal y como ocurriría en el caso anterior, por el planificador Metric, puesto que Optic registra 6 segundos aproximadamente frente a 0.59 segundos. En último lugar, cuando el problema alcanza su máximo tamaño los tiempos se exceden significativamente en el planificador LPG. Por otro lado, Optic también sufre un aumento aunque en menor medida. Todo esto nos inclina por escoger Metric como el mejor de los planificadores empleado, ya que obtiene un tiempo ínfimo comparado con sus competidores.

Table 2: Evolución temporal al variar el tamaño del problema

Tiempo(s)	**Metric**	**Optic**	**LPG**
Pequeño	0.11	0.88	0.58
Mediano	0.59	6.09	53.76
Grande	2.53	46.25	1211.52

En definitiva, observamos como el tiempo de resolución aumenta conforme aumentamos el tamaño del problema pero sería conveniente averiguar cuál es el recurso que implica este incremento del coste temporal. En otras palabras, deseamos identificar el factor, como podría ser el número de muelles o de contenedores, que influye principalmente sobre el tiempo de resolución. Para realizar este experimento, partiremos del problema mediano definido anteriormente. Sobre este problema aplicaremos cambios para llevar a cabo la evaluación comentada previamente. Antes de seguir avanzando, es necesario destacar que se va a emplear el planificador Metric, ya que el que mejores tiempos ha presentado en la anterior experimentación.

Comenzaremos, ampliando el número de muelles mientras mantenemos fijo el resto de elementos presentes en el puerto mediano. Esto, además, supone el incremento del número de cintas de transporte, grúas, así como el número de pilas. Más concretamente, se añadirán tantas cintas sean

necesarias para que todos los muelles estén conectados. Tal y como se aprecia en la Figura 15, podemos concluir que el número de muelles, así como el de cintas y sus objetos asociados, tiene un incremento del tiempo de resolución en el plan. Esto puede deberse a que aumenta el numero de instancias y por tanto, el espacio de búsqueda crece haciendo que el problema sea más complicado de resolver ya que tiene más acciones que explorar.

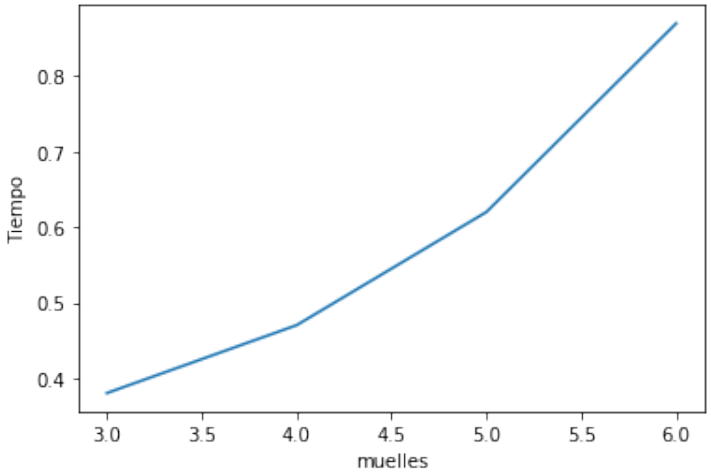

Figure 15: Evolución temporal al aumentar el numero de muelles

Como aumentar el número de muelles implica incrementar el número de grúas, cintas y pilas; solo nos falta evaluar cómo influye la cantidad de contenedores presentes en el puerto. Para ello, realizamos una experimentación similar a la anterior construyendo la Figura 16.

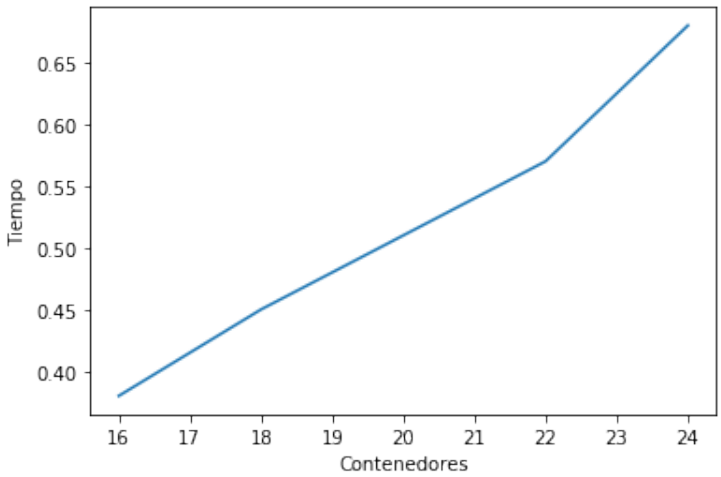

Figure 16: Evolución temporal al aumentar el numero de contenedores

Por último, solo nos falta evaluar cómo afecta sobre la construcción del plan el aumento del número de contenedores objetivos que deseamos. En este caso, a diferencia de los anteriores análisis, se ha empleado el planificador Optic, puesto que Metric no conseguía resultados en un tiempo razonable. Entonces, realizando un procedimiento similar a los anteriores casos, hemos ido aumentando el número de contenedores objetivos del problema. Con cinco contenedores objetivos obtenemos un tiempo de 4.07 segundos, mientras que aumentando el número a siete objetivos se alcanza el primer plan a los 5.63 segundos. Por otro lado, al aumentar los objetivos a 9 (el número máximo de contenedores que pueden estar en un muelle) el tiempo excede los límites. Todo esto nos sugiere que el cuello de botella más relevante en nuestro dominio sería el número de contenedores objetivos que se demandan.

Por último, destacar que si quisiéramos ejecutar un problema con un número de contenedores de objetivos que sea mayor estricto a la capacidad del muelle (entendida como nº pilas * altura de las pilas) el problema sería considerado como irresoluble por los planificadores.

3 Dominio Temporal

Para el desarrollo del dominio temporal se propone realizar una adaptación del dominio original cumpliendo las siguientes restricciones:

- Los contenedores deben de tener un peso, el tiempo de apilar/desapilar pasará a ser proporcional al peso de la misma y a la altura a la que se encuentra el contenedor. Es decir, a menor altura más tiempo
- Las cintas transportadoras tendrán un tiempo asociado para el transporte de una caja de un muelle a otro.
- El tiempo de colocar y recoger un contenedor de la cinta será proporcional al peso de dicho contenedor.

Por esta razón será necesario realizar unas modificaciones sobre el dominio original, especificando por ejemplo la duración de cada acción, o modificando el comportamiento de algunas reglas.

3.1 Modificaciones sobre el dominio proposicional

3.1.1 Predicados adicionales

Para la consecución de los objetivos comentados anteriormente ha sido necesario definir dos nuevos predicados:

- **inicioCinta ?c** - contenedor **?t** - cinta

  El predicado inicioCinta nos permite modelar un comportamiento para señalar que un contenedor está al inicio de la cinta.

- f**inalCinta ?c** - contenedor **?t** - cinta

  El predicado finalCinta nos permite modelar cuando un contenedor se encuentra al final de la cinta.

3.1.2 Definición de funciones

- (**peso ?c1** - contenedor) Esta función nos permite modelar el peso de un contenedor
- (**conaltura ?a1** - nivel) Mediante esta función podemos modelar de una forma numérica la altura de un determinado nivel
- (**tiempoTransporte**) Mediante la constante tiempoTransporte conseguimos modelar el tiempo que tarda en transportar una cinta un contenedor entre dos muelles.

3.1.3 Transportar un contenedor a otro muelle

A la hora de paralelizar las acciones se ha visto que era necesario tener una regla mover un paquete por la cinta.

```
(:durative-action mover
    :parameters (?c - contenedor ?mOrigen - muelle ?mDestino - muelle ?t - cinta)
    :duration (= ?duration (tiempoTransporte))
    :condition (and
        (at start (inicioCinta ?c ?t)) ; solo al comienzo debe comprobarse que el contenedor se encuentra al inicio de la cinta
        (over all (en ?c ?t)) ; el contenedor debe estar en la cinta durante todo el trayecto
        (over all (conecta ?t ?mOrigen ?mDestino)) ; en todo momento se debe comprobar la conexion entre muelles
    )
    :effect (and
        (at end (finalCinta ?c ?t)) ; al final de la accion el contenedor estara al final de la cinta
        (at end (not (inicioCinta ?c ?t))) ; al final no puede estar al principio de la cinta
    )
)
```

Figure 17: Regla mover

Tal y como se puede observar en la figura 17 la regla tiene una duración establecida por la constante **tiempoTransporte**. Además, se exigen como condiciones que al principio de la ejecución de la acción el contenedor esté al inicio de la cinta y durante la realización de la acción el contenedor se mantenga en la cinta y exista una cinta que conecta ambos muelles. Por último, el efecto será que al final de la acción el contenedor pasará a estar al final de la cinta.

3.2 Modificaciones sobre el problema

```
;pesos de contenedor
(= (peso c1) 5)
(= (peso c2) 2)
(= (peso c3) 3)
(= (peso c4) 6)
(= (peso c5) 5)
(= (peso c6) 5)
(= (peso c7) 5)
(= (peso c8) 5)
(= (peso c9) 10)
(= (peso c10) 8)
(= (peso c11) 5)
;alturas pilas
(= (conaltura n0) 1)
(= (conaltura n1) 2)
(= (conaltura n2) 3)
(= (conaltura n3) 4)
;tiempo de transportar un contenedor en la cinta
(= (tiempoTransporte) 10)
```

Figure 18: Modificaciones sobre el problema del dominio temporal

A la hora de adaptar el problema del dominio temporal se necesita, tal y como se puede observar en la figura 18 definir las funciones que se comentaron en el apartado de Definición de funciones. De esta forma especificamos el peso de cada uno de los contenedores, la altura numérica de cada nivel, y el tiempo que tarda una cinta en transportar un paquete.

Además, dado que ahora queremos minimizar el tiempo de ejecución global del problema tenemos que especificarlo, añadiendo la siguiente línea a nuestro problema: (:**metric minimize (total-time)**)

3.3 Experimentación y resultados

En este apartado se presenta la experimentación llevada a cabo sobre el dominio temporal que hemos descrito a lo largo de la memoria. En este caso, se probará sobre LPG y Optic la definición de nuestro dominio sobre el problema inicial planteado en el boletín del ejercicio. Sin embargo, solo se han obtenido resultados con el planificador LPG, aunque en un tiempo elevado. Por lo visto, la búsqueda de la solución cuando aplicábamos Optic excedía el tiempo y no llegaba alcanzar ningún plan. Debido a esto, la Tabla 3 expone los resultados que proporciona LPG con diferentes configuraciones del estado inicial. Más concretamente, en función del número de grúas presentes en cada muelle. Vemos que si hubiera más contenedores, el número de grúas en cada muelle podría suponer un mayor paralelismo, es decir, múltiples acciones ejecutándose simultáneamente, lo que provocaría planes de menor duración.

Table 3: Evaluación del dominio temporal

#Gruas en el Muelle1	#Grúas en el Muelle 2	Duración plan
1	1	38.917
1	2	35.667
2	2	35.667

Este incremento del coste temporal, que incluso ha provocado que Optic no obtenga resultados, se debe sin duda al gran abanico de variabilidad que aporta el eje temporal en la construcción de un plan, ya que se introduce una mayor complejidad en la búsqueda. Se intentaron diversas formas de optimizar nuestro dominio pero ninguna llegó a buen puerto.

4 Dominio Númerico

En este ejercicio se pretende asentar el empleo de los fluents en PDDL estudiados en clase. Por ello, se propone modificar el dominio temporal con el objetivo de introducir una variable numérica que represente el consumo de algún recurso. Además, tal y como indica el boletín, este consumo deberá ser inversamente proporcional al consumo de tiempo. En otras palabras, cuanto menos tiempo transcurra más recurso debemos consumir. Antes de abordar el recurso escogido, es necesario conocer que ahora podremos resolver el problema minimizando en términos de éste.

4.1 Consideración de la gasolina como recurso

Hemos decidido considerar como recurso numérico la gasolina necesaria para el funcionamiento de las cintas transportadoras. Además, estas cintas dispondrán de dos posibles modos de transporte: lento y rápido; cada uno de ellos con un consumo acorde al tiempo transcurrido para el transporte del contenedor a otro muelle. Por otro lado, se debe implementar una versión del dominio donde la gasolina no sea renovable, es decir, que no se pueda repostar el depósito. Una vez implementada esta versión, construiremos una versión renovable aplicando las modificaciones necesarias.

4.2 Modificaciones sobre el dominio temporal

Antes de adentrarnos en detalle sobre las modificaciones realizadas, es necesario indicar que en los sucesivos apartados se va a mencionar todo aquello que haya sido necesario incorporar para lograr modelar tanto la versión no renovable como la versión en la que podemos repostar.

4.2.1 Funciones adicionales

Para poder realizar los objetivos planteados se han añadido, en primera instancia, las siguientes funciones:

- (tiempoTransporteRapido): Con esta función logramos establecer el tiempo que tarda la cinta en transportar un contenedor a otro muelle en modo rápido.
- (tiempoTransporteLento): Igual que en el caso anterior pero en el modo lento de transporte.
- (consumoTransporteRapido): Sería la cantidad de gasolina consumida por una cinta cuando se utiliza el transporte en su modo rápido.
- (consumoTransporteLento): Análogo al anterior pero en el caso de un transporte a velocidad lenta.
- (gasolinaDisponible): Esta función nos permite conocer los litros de gasolina que disponemos en un momento dado.
- (tamañoDeposito): Mientras que con esta función establecemos la capacidad total del depósito de gasolina. Será empleada a la hora de repostar para conocer nuestros límites.
- (tiempoRespostar): Hace referencia al coste temporal que supone repostar el depósito.

4.2.2 Rediseño de operadores

La incorporación de este recurso supone, como cabría esperar, modificar el modelado de la acción relacionada con el transporte de un contenedor desde el inicio al final de la cinta. Estamos hablando, más concretamente, del operador mover que comentamos en anteriores apartados. De forma que para poder diferenciar entre un consumo u otro es necesario distinguir cuando se está realizando un transporte lento o rápido. Esto nos obliga a desglosar dicho operador en dos operadores: uno encargado del transporte en el modo rápido y otro para el lento. Su implementación PDDL puede observarse en la Figura 19.

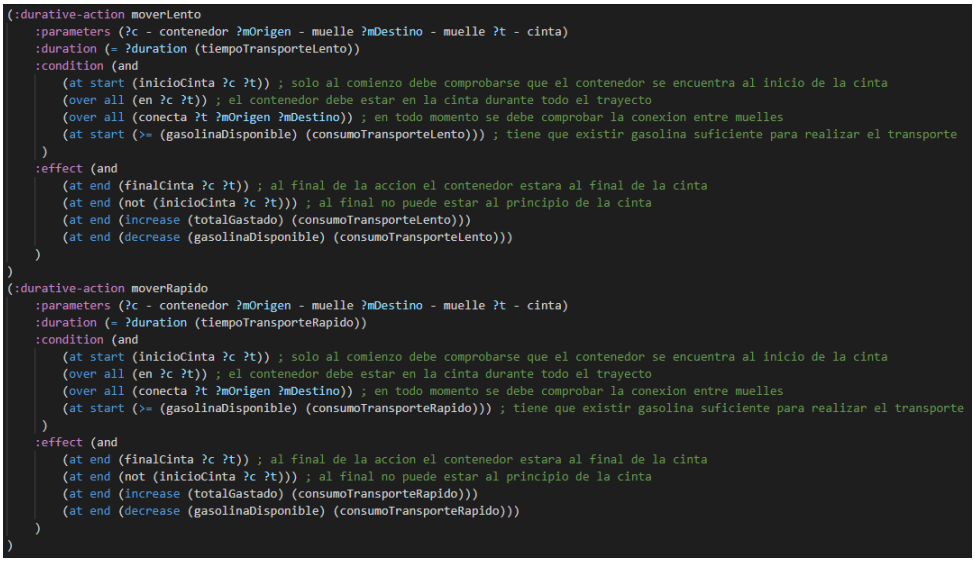

Figure 19: Desglose del operador mover al incorporar la gasolina como recurso numérico

La diferencia principal estriba cuando definimos la duración del operador, así como en la premisa que comprueba si existe suficiente gasolina para poder realizar la acción en cuestión. Respecto a los efectos, cada operador consume del depósito la cantidad apropiada de combustible y se va registrando la cantidad total de gasolina empleada a lo largo de toda la planificación. Esto nos será útil para poder optimizar el plan en términos del combustible consumido. En todos los casos, esta modelización ha sido posible gracias a las diferentes consultas sobre las funciones que hemos comentado previamente.

Por otro lado, en el caso de que nos encontremos en la versión renovable del dominio, es necesario construir un nuevo operador encargado de repostar, hasta su capacidad máxima, el depósito de gasolina. Tal y como se puede observar en la implementación PDDL que se muestra en la Figura 20, la condición para que se instancie este operador es que el nivel de gasolina disponible sea inferior al consumo de un transporte lento, ya que esta es la acción más barata del dominio. En otras palabras, si no es capaz de realizar el transporte de menor consumo no podrá realizar nada más hasta que no rellenemos de nuevo el depósito. En cuanto al resto del código, solo resta comentar que para la duración de este operador empleamos uno de los predicados comentados anteriormente.

```
(:durative-action repostar
    :parameters ()
    :duration (= ?duration (tiempoRepostar))
    :condition (and
        (at start (< (gasolinaDisponible) (consumoTransporteLento))) ; si no tenemos gasolina suficiente ni para
    )                                                                 ; hacer el transporte mas barato...
    :effect (and
        (at end (= (gasolinaDisponible) (tamañoDeposito))) ; repostamos hasta su capacidad maxima
    )
)
```

Figure 20: Operador para repostar el depósito de gasolina

4.3 Modificaciones sobre el problema

Pero todo esto esto supone añadir información a la hora de definir el problema para que nuestro dominio pueda resolver e instancias sus nuevas funcionalidades. Más concretamente, será necesario introducir los valores numéricos sobre todas las funciones que hemos incorporado y comentado en anteriores secciones. De forma que siguiendo una cierta coherencia con lo que plantea el enunciado hemos implementado los valores tal y como se muestran en la Figura 21.

```
(= (tiempoTransporteLento) 10)
(= (tiempoTransporteRapido) 5)
(= (consumoTransporteLento) 50)
(= (consumoTransporteRapido) 100)
;
(= (gasolinaDisponible) 400)
(= (totalGastado) 0)
(= (tamañoDeposito) 400)
(= (tiempoRepostar) 20)
```

Figure 21: Implementación de las nuevas *functions* en el problema numérico

Por otro lado, ya podemos indicar que deseamos resolver el problema minimizando el consumo de combustible, tal y como se expone en la Figura 22. Aunque esto no quiere decir, tal y como veremos más adelante, que no podamos minimizar a su vez el tiempo del plan.

```
;metrica a minimizar
(:metric minimize (totalGastado))
)
```

Figure 22: Métrica que indica que se desea minimizar el combustible consumido durante el plan

4.4 Experimentación y evaluación

En esta experimentación solamente se realizarán pruebas con el planificador LPG. Tal y como sugiere el boletín, se llevarán a cabo pruebas para dos dominios numéricos: uno el que se permite renovar o repostar la gasolina y otro en el que no es posible. Para ello, definiremos, en primer lugar, dos instancias de problemas:

- Instancia A: se se proponen una serie de valores razonables respecto al peso de los contenedores, el consumo de gasolina de las cintas (en su versión rápida y lenta) y el tiempo de repostar (en el caso en el que se permita).
- Instancia B: se establecen distintos valores a los parámetros anteriormente citados donde, en su mayoría, adquirirán un mayor peso, mayor consumo y mayor tiempo de respuesto.

En la tabla 4, se muestran los resultados obtenidos cuando optimizamos el plan respecto al recurso introducido en el dominio, es decir, la gasolina; mientras la Tabla 5 expone la duración de los planes obtenidos cuando minimizamos solamente dicho factor. En primer lugar, determinamos que la instancia B provoca un aumento en el consumo de gasolina. Por otra parte, observamos que cuando el dominio es renovable sorprendentemente cuando nos ceñimos a minimizar el tiempo se produce un aumento de la duración del plan respecto a cuando minimizamos la gasolina. En el caso no renovable, se refleja un comportamiento inverso y más coherente.

Table 4: Resultados obtenidos minimizando el consumo de gasolina

Domino Empleado	**Instancia del problema**	**Duración / Coste**
No Renovable	A	37.083 / 150 litros
No Renovable	B	84.83 / 200 litros
Renovable	A	30.5 / 150 litros
Renovable	B	81.25 s / 200 litros

Table 5: Resultados obtenidos minimizando la duración del plan

Domino Empleado	**Instancia del problema**	**Duración / Coste**
No Renovable	A	32.08
No Renovable	B	81.38
Renovable	A	42.33
Renovable	B	140.58

5 Desarrollo parcial de un árbol POP

Normalmente una planificación state-space, donde la búsqueda se realiza explorando el espacio de los posibles estados y decidiendo en cada momento entre múltiples acciones; proporciona planes secuenciales que, en ocasiones, distan de ser eficientes. Debido a esto, puede resultar conveniente construir un plan bajo el marco de la planificación plan-space de forma que podamos construir planes que permitan la aplicación de varias acciones en paralelo y, solo en caso de necesidad, se establecerá un orden entre algunas de ellas. Ahora cada uno de los nodos de nuestro árbol de búsqueda será un plan parcial y tendremos que decidir en cada momento qué flaw queremos cumplir. En función de nuestra elección, iremos introduciendo acciones y restricciones sobre nuestro plan parcial. Cabe destacar que la planificación desde esta perspectiva se realiza mediante una búsqueda backward, es decir, partiendo desde los objetivos planteados en el problema en cuestión hasta alcanzar el estado inicial. Más concretamente, estamos hablando de la construcción de un árbol POP (Partial-Order Planning).

En este ejercicio desarrollaremos tres iteraciones de un árbol POP enfocado al problema reducido que se muestra en la Figura 23, donde nuestros objetivos serían tener los contenedores c1, c3 y c4 en el muelle 1 y disponibles para su recogida.

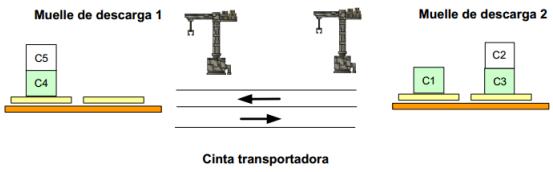

Figure 23: Problema reducido del puerto

En primer lugar, mostraremos el árbol POP de forma general, tal y como sugiere la Figura 24. Posteriormente, describiremos en más detalle cuál ha sido el desarrollo de éste. Antes de nada, comentar que se ha comenzado resolviendo un objetivo para después resolver la instanciación de una variable y, ya en último lugar, se ha optado por resolver un sub-objetivo. Esta selección de flaws se ha realizado aleatoriamente. Aunque cabe destacar que en las clases teóricas se ha estudiado una heurística de selección que sugiere escoger primero aquellos flaws cuya resolución implique el menor número de alternativas posibles. De esta forma, reducimos considerablemente la ramificación y la amplitud de la búsqueda.

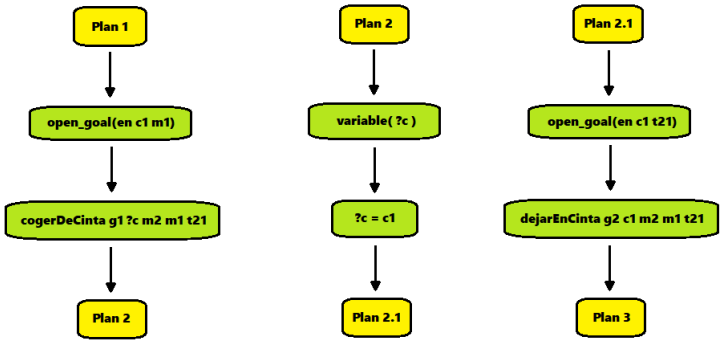

Figure 24: Esquema general del Árbol POP construido

En primer lugar, partiremos del Plan 1 (expuesto en la Figura 25) en el que reflejamos el estado inicial comentado anteriormente y el estado final con los objetivos que se solicitan. Además, tal y como se observa en la figura, existiría una relación de orden entre estos dos estados (flecha discontinua).

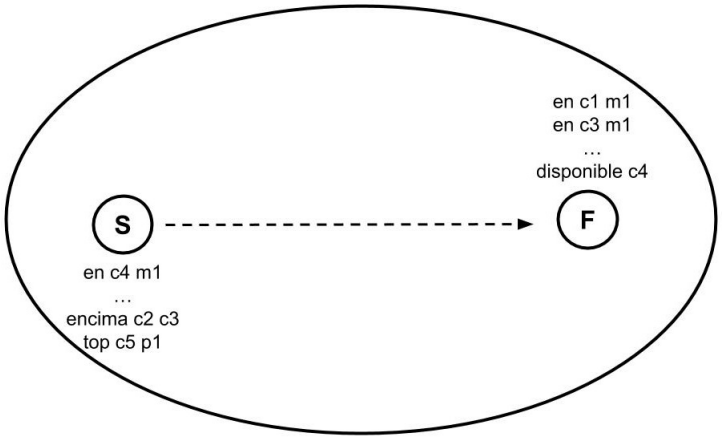

Figure 25: Representación del nodo inicial del árbol POP, el plan 1

En este caso, se presentan los siguientes flaws:

- Open goals: en c1 m1, en c3 m1, en c4 m1, disponible c1, disponible c3, disponible c4

Entonces, siguiendo la filosofía backward, decidimos resolver el objetivo en c1 m1. Para poder conseguir este objetivo sólo existe una alternativa: coger con la grúa del muelle 1 dicho contenedor desde la cinta que conecta ambos muelles. Según nuestro modelado, es en ese momento cuando el contenedor adquiere la nueva ubicación. De esta forma obtendríamos el siguiente nodo que muestra la Figura 26. Este nuevo elemento, implica que definamos una relación causal entre la acción que recoge el contenedor de

la cinta y el estado final, ya que dicha acción logra proporcionar una de las premisas que requiere nuestro estado objetivo. Además, deberá existir una relación de orden entre el estado inicial y esta acción, tal y como refleja la conexión discontinua.

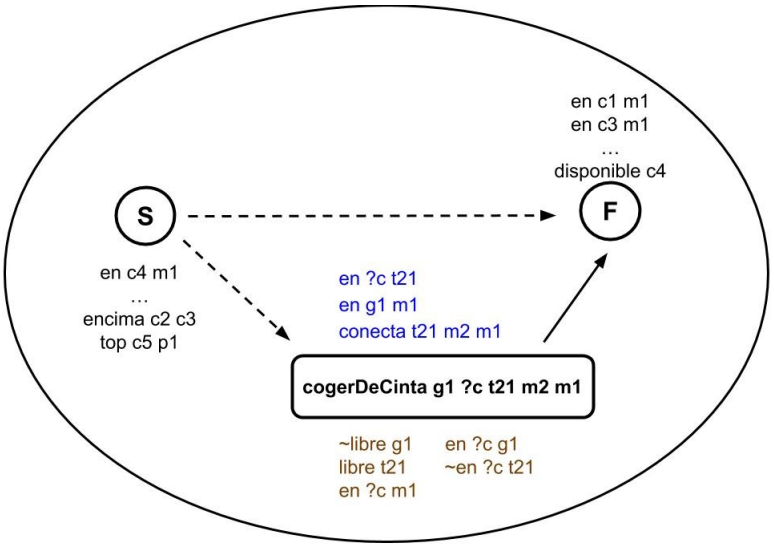

Figure 26: Representación del nodo inicial del árbol POP, el plan 2

A partir del plan 2, obtenemos los siguientes flaws:

- Open goals: en ?c t21, en g1 m1, conecta t21 m2 m1
- Variables: ?c

Al final decidimos resolver la instanciación de la variable ?c referente al contenedor que se encuentra sobre la cinta t21 de nuestro puerto. Esta variable podría instanciarse con cualquiera de los contenedores definidos pero, en este caso, debe asignarse con el contenedor c1 para que genere el efecto deseado. De esta manera, obtenemos el plan 2.1 que muestra la Figura 27

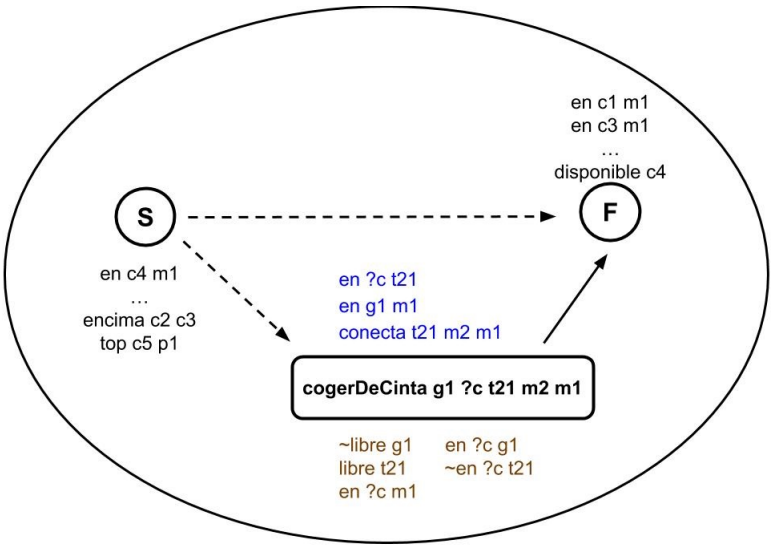

Figure 27: Representación del plan 2.1

Respecto al plan 2.1 que acabamos de obtener, se generan los siguientes flaws:

- Open goals: en c1 t21, en g1 m1, conecta t21 m2 m1

Entonces, en último lugar, ya que esta sería la tercera iteración, decidimos resolver el sub-objetivo en c1 t21, puesto que el resto de flaws tendrían una resolución más sencilla que implicaría la conexión con el estado inicial simplemente. Respecto al sub-objetivo seleccionado, nuevamente sólo disponemos de una posible alternativa para resolverlo: dejar en la cinta el contenedor c1 utilizando para ello la grúa del muelle 2. Este paso, además, implica cambios en las relaciones causales y de orden, tal y como reflejamos en la Figura 28.

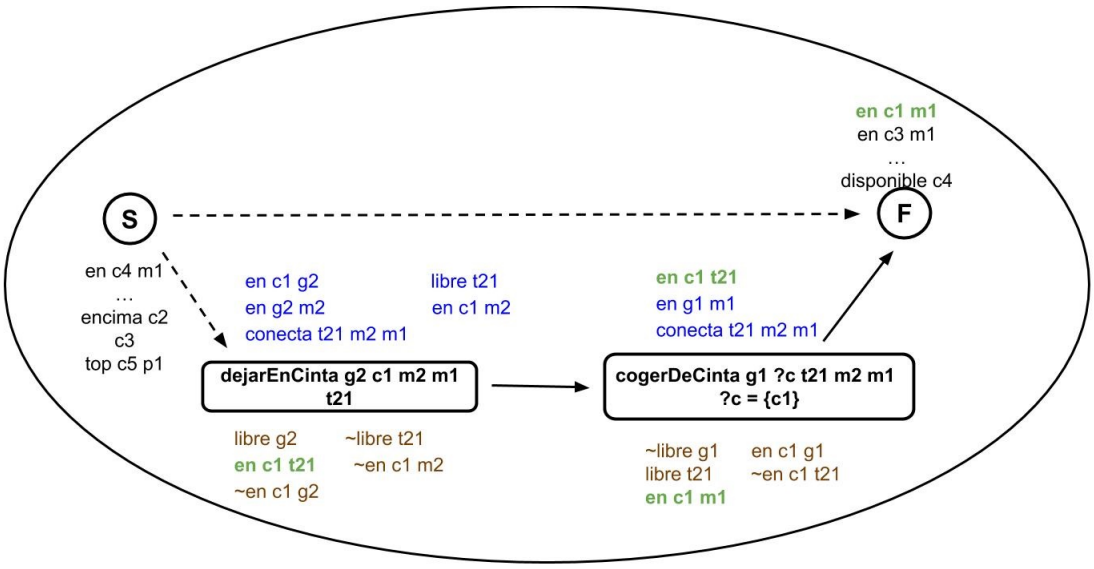

Figure 28: Representación del plan 3

Y con todo esto finalizamos el ejercicio dejando los siguientes flaws:

- Open goals: en c1 g2, en g2 m2, conecta t21 m2 m1, libre t21, en c1 m2

Si continuáramos en el desarrollo de nuestro Árbol POP, observaríamos que al final sólo se podrían paralelizar aquellas acciones realizadas por grúas distintas o, más bien, en muelles distintos. En otras palabras, todas las acciones en un mismo muelle deberán decidir, por necesidad, una relación de orden. Y, respecto a las acciones que involucren recoger un contenedor de la cinta transportadora, será necesario un cierto orden o causalidad de modo que la otra grúa haya dejado dicho contenedor sobre la cinta previamente.

6 Graphplan

Una forma de encontrar la solución a nuestro problema sería aplicar lo que se denomina una búsqueda ciega, la cual estaría basada en una exploración exhaustiva de todo el espacio de búsqueda hasta encontrar un estado que satisfaga nuestros objetivos. No obstante, esta metodología es inviable cuando se trata de resolver problemas de una cierta envergadura. Debido a esto, surge la necesidad de guiar nuestro proceso de búsqueda de soluciones de forma que podamos acelerarlo y seguir obteniendo soluciones de buena calidad. Estamos hablando de aquellos algoritmos de búsqueda que incorporan una heurística en sus cálculos. Antes de seguir avanzando, saber que una heurística consiste en una función capaz de estimar cuál sería el coste de alcanzar los objetivos planteados desde el estado en el que nos encontramos.

De esta manera, conseguimos dotar de cierta información al algoritmo de búsqueda para que pueda dirigir la búsqueda, de forma eficiente, hacia nuestros objetivos. Aunque la heurística suele estar condicionada por el problema al que nos enfrentamos, no podemos calcularla teniendo en cuenta todas las restricciones que presenta dicho problema, ya que sería igual de costoso que resolver el problema original. En otras palabras, para poder calcular nuestra heurística es necesario que partamos del problema en una versión relajada. Más concretamente, tal y como se especifica en el enunciado del boletín, no tendremos en cuenta los efectos negativos de los operadores y no contemplaremos las relaciones de exclusión mutua. Además, puesto que el problema planteado inicialmente es demasiado extenso, nos basaremos en el problema reducido que presentamos en el anterior apartado (Figura 23). Por otra parte, indicar que el grafo de planificación relajado propuesto para el ejercicio se encuentra en el fichero Excel adjunto en la entrega del proyecto. Sin más dilación, resolveremos las preguntas planteadas en el boletín:

1-¿Cuál es el primer nivel en el que se iniciaría la etapa de extracción de un plan en Graphplan?

Una vez se han alcanzado todos los objetivos planteados tras desarrollar las necesarias capas de Proposición y Acción, es hora de extraer el plan relajado. Para ello, hay que recorrer el Graphplan en el orden inverso a su construcción. En otras palabras, partimos del último objetivo alcanzado y, por lo tanto, empezaremos en el último nivel de Proposición que hayamos definido.

2-Calcula el valor el valor de las heurísticas hmax y hsum para los tres objetivos definidos.

Tal y como se puede observar en el fichero Excel adjunto, tras la construcción del grafo de planificación relajado hemos obtenido los siguientes valores:

- **hmax = 4**, puesto que el último de los objetivos se consigue en ese nivel.
- **hsum = 9**, siendo el resultado de acumular los niveles donde se alcanzan todos los objetivos

3-Extrae un plan relajado para los tres objetivos sobre el grafo de planificación relajado. Mostrar la extracción del plan relajado y demostrar cómo este plan se puede extraer en tiempo polinómico y sin necesidad de operaciones de backtracking

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 que objetivos de han alcanzado en dicho nivel y apuntamos las acciones que han permitido alcanzarlos (destacadas en rojo en el fichero Excel). A su vez, estas acciones requieren de ciertas premisas que habrán sido generadas por

acciones en un nivel. De esta manera, vamos tirando del hilo hasta llegar al estado inicial del problema. Cabe destacar que es posible encontrarnos con otros objetivos por el camino, lo que conlleva abrir otro ruta. Respecto al coste del algoritmo, vemos que es exponencial puesto que hay que explorar toda posible instanciación en función de los estados que estemos contemplando en cada instante. Esto dependerá del número de operadores que hayamos definido, así como del número de parámetros que requieran éstos. Una vez alcanzado todos los objetivos, se trata de realizar el procedimiento comentado anteriormente. En el caso de que deseemos obtener un plan solución real, si que sería necesario aplicar la técnica de backtracking, tal y como se sugirió en las clases teóricas.

4- ¿Cuál de las tres heurísticas calculadas (hsum, hmax, plan relajado) es la más informada para este problema? ¿Por qué?

Consideramos la heurística **hsum** como la más informada, puesto que contempla en qué nivel se alcanza cada uno de los objetivos planteados en el problema relajado, mientras que la **hmax** sólo 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), concluimos que su estimación sería más pobre que la **hsum**.

Por último, llama la atención que el plan relajado finaliza con uno de los contenedores sostenido por la grúa del muelle destino. Esto se debe a que al no borrar estados a lo largo del desarrollo del Graphplan el contenedor nunca ha dejado de estar disponible desde que se consiguió en un nivel anterior.

7 Conclusiones

Para la realización de este trabajo se ha realizado una codificación de un dominio proposicional en PDDL. Para ello, el primero de los pasos es conocer en profundidad los detalles de nuestro dominio, de forma que podamos realizar una codificación de las distintas acciones y de los distintos predicados que envuelven al problema del puerto. Por otro lado, para la correcta codificación del problema y que funcione adecuadamente en todos los planificadores ha sido necesario utilizar los conocimientos compartidos en las clases de teoría. Más concretamente, algunos planificadores no admiten precondiciones negadas, por lo que ha sido necesario utilizar otras técnicas para suplir esta carencia.

Por otra parte, concluir que uno de los aspectos más relevantes a la hora de modelar un dominio viene a ser la experimentación realizada durante el desarrollo. De esta forma, seremos capaces de identificar los puntos débiles

de nuestro modelado, como pudiera ser un número excesivo de parámetros lo que implicaría una demora del tiempo de resolución o, por otro lado, poder determinar qué elementos del dominio son cuellos de botella. En definitiva, conocer los límites de nuestra simulación.

En último lugar, se ha ido construyendo de forma incremental un dominio cada vez más complejo, es decir, introduciendo el factor temporal así como el consumo de recursos. Además, tras cada incorporación, se llevan a cabo una serie de experimentaciones para poder ir extrayendo conclusiones a medida que avanzamos en el desarrollo del dominio. De este modo, hemos aprendido una metodología adecuada para el modelado de dominios dentro del ámbito de la planificación inteligente.