Dominio Terminal de Aeropuerto

Jorge Juan González; Arlenis Ching Suárez

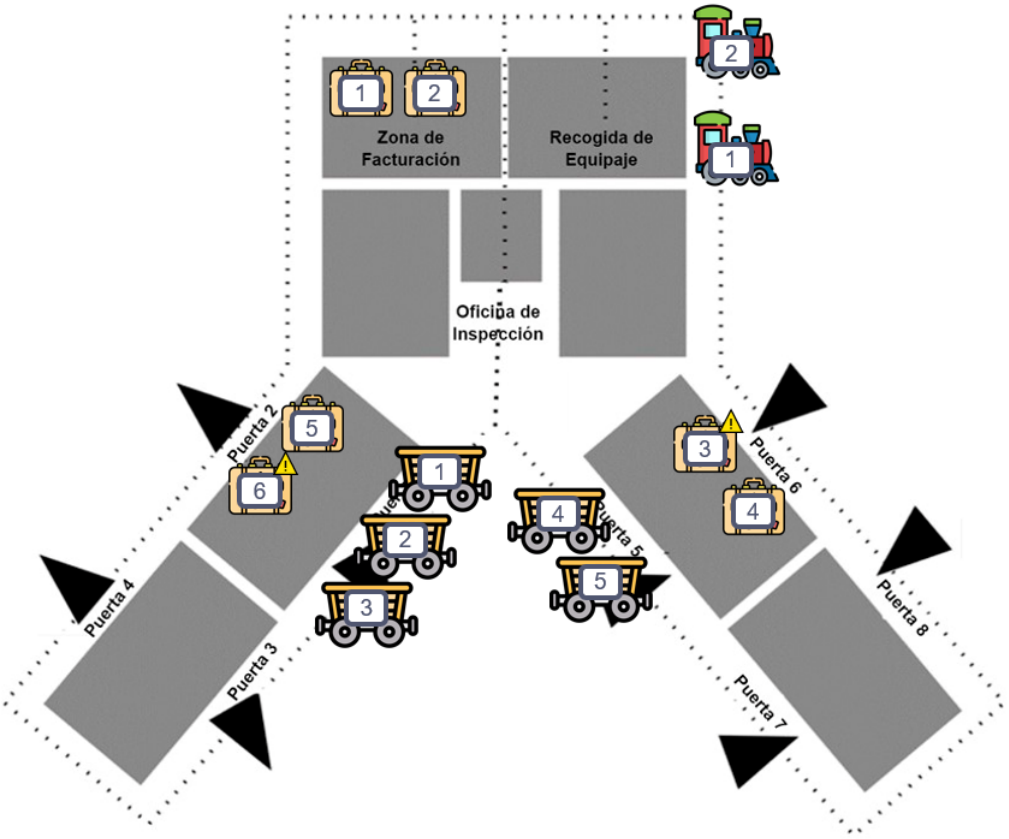
14 de febrero de 2022

Índice

1. Introducción al dominio

El dominio proposicional del aeropuerto consiste en trenes, compuestos de máquinas tractoras y vagonetas conectadas a ellas, sobre los que se pueden cargar equipajes. Los trenes pueden moverse entre las distintas localizaciones del aeropuerto.

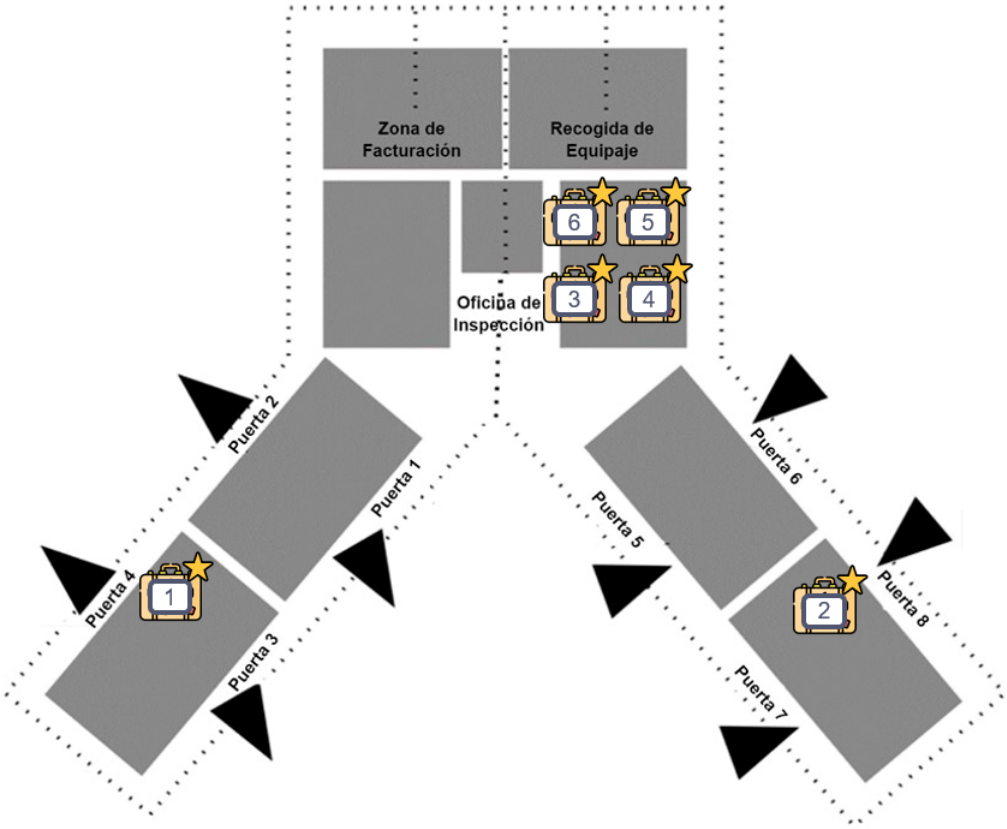
Figura 1: Problema propuesto.



Tal y como se puede ver en la Figura 1, un aeropuerto puede tener distintas localizaciones donde puede haber máquinas, vagonetas y equipajes. Algunos equipajes pueden estar marcados como peligrosos y deberán ser analizados en la Oficina de Inspección del aeropuerto antes de ser llevados a su destino.

El objetivo final es dejar todos los equipajes en sus correspondientes destinos, habiendo analizado aquellos que sean peligrosos. Por ejemplo, una vez resuelto el problema mostrado en la Figura 1, el estado final objetivo sería el que se muestra en la Figura 2.

Figura 2: Objetivo propuesto.



2. Dominio proposicional

A continuación se describen los diferentes tipos, predicados y acciones utilizados en nuestra implementación del dominio del aeropuerto.

2.1. Tipos

- **Localización:** Puntos del aeropuerto como puertas de embarque o zonas de recogida o facturación de equipaje. La localización .ºficina de Inspección"será una constante.
- **Equipaje:** Objeto que puede ser cargado en una vagoneta y que tiene un destino al que debe ser transportado.
- **Vagoneta:** Vagoneta con capacidad limitada en la que se transportan los equipajes.
- **Máquina:** Vehículo tractor que mueve el tren a las distintas localizaciones del aeropuerto.
- **Número:** Tipo que podemos utilizar para contar sin usar operadores aritméticos. El número cero será una constante.

2.2. Predicados

- (**adyacent ?loc** - localizacion, **?loc2** - localizacion)
  El predicado **adyacent** indica que dos localizacion son adyacentes, es decir, que una máquina puede viajar de una localización a otra.
- (**at ?maq** - maquina **?loc** - localizacion)
  El predicado **at** indica que una máquina se encuentra en una localización.
- (**connected ?vag** - vagoneta **?v** - (either vagoneta maquina))
  El predicado **connected** indica que una vagoneta está conectada directamente detrás de una vagoneta o una máquina.

- (**follow ?v** - (either vagoneta maquina) **?maq** - maquina)
  El predicado **follow** indica que una vagoneta forma parte del tren que dirige una máquina. Una máquina siempre deberá "seguirse.ª sí misma.

- (**last ?maq** - maquina **?v** - (either vagoneta maquina))
  El predicado **last** indica que una vagoneta es la última vagoneta conectada. En el caso de que una máquina no tenga vagonetas conectadas, el último vehículo detrás de la máquina será la propia máquina.

- (**occupied ?vag** - vagoneta **?eq** - equipaje)
  El predicado **occupied** indica que un equipaje se encuentra cargado dentro de una vagoneta.

- (**danger ?eq** - equipaje)
  El predicado **danger** indica que un equipaje es peligroso y debe ser analizado en la oficina de inspección.

- (**notdanger ?eq** - equipaje)
  El predicado **notdanger** indica que un equipaje no es peligroso y puede ser entregado en su destino.

- (**destination ?eq** - equipaje **?loc** - localizacion)
  El predicado **destination** indica que un equipaje debe ser depositado en una localización concreta.

- (**waiting ?x** - (either equipaje vagoneta) **?loc** - localizacion)
  El predicado **waiting** indica que una vagoneta (vacía y desconectada) o un equipaje (no cargado en ninguna vagoneta) se encuentran en una localización. Se puede interpretar como que un equipaje está .esperando.ª ser cargado o que una vagoneta está .esperando.ª ser conectada.

- (**toanalyce ?eq** - equipaje)
  El predicado **toanalyce** indica que un equipaje peligroso se encuentra depositado en la oficina de inspección en vista de ser analizado y marcado como seguro.

- (**done ?eq** - equipaje)
  El predicado **done** indica que un equipaje ha sido entregado en su destino.

- (**next ?n1** - numero **?n2** - numero)
  El predicado **next** indica el orden entre dos números.

- (**carried ?vag** - vagoneta **?n** - numero)
  El predicado **carried** indica el número de equipajes cargados en una vagoneta.

Figura 3: Predicados el ejercicio 1.

```
(:predicates
    (adyacent ?loc1 - localizacion, ?loc2 - localizacion) 1👁  ;Una localización es adyacente a otra
    (at ?maq - maquina ?loc - localizacion)  ;Una máquina se encuentra en una localización
    (connected ?vag - vagoneta ?v - (either vagoneta maquina)) 1👁 1☑ 1☒ ;Una vagoneta (?vag) sigue a otro vehículo (?v)
    (follow ?v - (either vagoneta maquina) ?maq - maquina) 6👁 1☑ 1☒ ;Un vehículo (?v) pertenece al tren que dirige una máquina
    (last ?maq - maquina ?v - (either vagoneta maquina)) 2👁 2☑ 2☒ ;Un vehículo (?v) es el último eslabón del tren que dirige una máquina
    (occupied ?vag - vagoneta ?eq - equipaje) 2👁 1☑ 2☒ ;Un equipaje ocupa un espacio dentro de una vagoneta
    (danger ?eq - equipaje) 2👁 1☒ ;Un equipaje es seguro
    (notdanger ?eq - equipaje)  ;Un equipaje es peligroso
    (destination ?eq - equipaje ?loc - localizacion) 1👁 1☒ ;Destino al que debe ser llevado un equipaje
    (waiting ?x - (either equipaje vagoneta) ?loc - localizacion) 2👁 2☑ 2☒ ;Un equipaje o vagoneta (?x) espera a ser recogido en una localización
    (toanalyce ?eq - equipaje) 1👁 1☑ 1☒ ;Un equipaje sospechoso está en la OficinaInspeccion para ser analizado
    (done ?eq - equipaje) 1☑ ;Un equipaje ha llegado a su destino y no es sospechoso
    (next ?n1 - numero ?n2 - numero) 3👁 ;Orden entre números siendo ?n1 menor que ?n2
    (carried ?vag - vagoneta ?n - numero) 3👁 3☑ 3☒ ;Cantidad de equipaje cargado por una vagoneta
)
```

2.3. Acciones

En este apartado explicaremos las diferentes acciones u operadores que permiten cambiar el estado actual del aeropuerto. Por cada acción explicaremos cuáles son sus precondiciones y qué cambios o efectos producen.

2.3.1. Mover una máquina

La primera acción se encargará de mover una máquina de una localización a otra. Estas localizaciones deben ser adyacentes y la máquina debe encontrarse en la localización de origen.
Como resultado de esta acción, la máquina pasará a encontrarse en la localizacion de destino.

Figura 4: Acción move.

```
(:action move
    :parameters (
        ?maq - maquina
        ?orig - localizacion
        ?dest - localizacion
    )
    :precondition (and
        (at ?maq ?orig) ;La máquina debe estar en el origen
        (adyacent ?orig ?dest) ;El origen y el destino deben ser adyacentes
    )
    :effect (and
        (not (at ?maq ?orig)) ;La máquina dejará de estar en el origen
        (at ?maq ?dest) ;La máquina estará en el destino
    )
)
```

2.3.2. Cargar un equipaje

La acción de cargar un equipaje requiere que una vagoneta siga a una máquina que a su vez se encuentra en la misma localización que el equipaje que se desea cargar. También se requiere que exista un número después del número actual de equipajes en la vagoneta, es decir, que no se haya alcanzado la capacidad máxima de la vagoneta.
Los efectos de esta acción son que el equipaje deja de estar a la espera de ser cargado, pasa a estar contenido dentro de la vagoneta y el número de equipajes dentro de la vagoneta pasa a ser el siguiente número después de ese.

Figura 5: Acción load.

```
(:action load
    :parameters (
        ?eq - equipaje
        ?vag - vagoneta
        ?maq - maquina
        ?quant - numero
        ?upperQuant - numero
        ?loc - localizacion
    )
    :precondition (and
        (at ?maq ?loc) ;La máquina debe encontrarse en la misma localización que el equipaje
        (follow ?vag ?maq) ;La vagoneta en la que se cargará el equipaje debe seguir a la máquina
        (waiting ?eq ?loc) ;El equipaje a cargar debe estar a la espera de ser cargado (fuera de cualquier vagoneta)
        (carried ?vag ?quant) ;Debe existir un número superior al número de equipajes cargados en la vagoneta
        (next ?quant ?upperQuant)
    )
    :effect (and
        (not(waiting ?eq ?loc)) ;El equipaje dejará de estar a la espera de ser cargado
        (occupied ?vag ?eq) ;El equipaje pasará a estar dentro de la vagoneta
        (not(carried ?vag ?quant)) ;La cantidad de equipajes en la vagoneta pasará a ser el número superior
        (carried ?vag ?upperQuant)

    )
)
```

2.3.3. Descargar un equipaje seguro

La acción de descargar un equipaje seguro requiere que el equipaje a descargar se encuentre cargado en una vagoneta conectada a una máquina que se en cuentra en la localización destino del equipaje. También se requiere que el equipaje esté marcado como seguro.
Esta acción hará que el equipaje deje de estar cargado en esa vagoneta, que sea marcado como entregado en su destino y que el número de equipajes dentro de esa vagoneta pasa a ser el número anterior.

Figura 6: Acción unload notdanger.

```
(:action unload_notdanger
    :parameters (
        ?eq - equipaje
        ?vag - vagoneta
        ?maq - maquina
        ?quant - numero
        ?lowerQuant - numero
        ?loc - localizacion
    )
    :precondition (and
        (notdanger ?eq) ;El equipaje debe ser seguro
        (destination ?eq ?loc) ;El destino del equipaje debe ser la localización en la que estamos
        (at ?maq ?loc) ;La máquina también se debe encontrar en esa localización
        (follow ?vag ?maq) ;La vagoneta que contiene al equipaje debe seguir a la máquina
        (occupied ?vag ?eq) ;El equipaje debe estar cargado en la vagoneta
        (carried ?vag ?quant) ;Debe existir un número inferior al número de equipajes cargados en la vagoneta
        (next ?lowerQuant ?quant)
    )
    :effect (and
        (done ?eq) ;El equipaje se marcará como entregado
        (not(destination ?eq ?loc)) ;El equipaje ya no tendrá un destino
        (not(occupied ?vag ?eq)) ;El equipaje ya no estará en la vagoneta
        (not(carried ?vag ?quant)) ;La cantidad de equipajes en la vagoneta pasará a ser el número inferior
        (carried ?vag ?lowerQuant)
    )
)
```

2.3.4. Descargar un equipaje peligroso

La acción de descargar un equipaje peligroso, al igual que en la acción anterior, requiere que el equipaje a descargar se encuentre cargado en una vagoneta conectada a una máquina, pero en este caso la máquina debe estar en la oficina de inspección. Además, el equipaje debe estar marcado como peligroso.
Esta acción hará que el equipaje deje de estar cargado en esa vagoneta, que sea marcado como pendiente de ser analizado y, de nuevo, que el número de equipajes dentro de esa vagoneta pasa a ser el número anterior.

Figura 7: Acción unload danger.

```
(:action unload_danger
    :parameters (
        ?eq - equipaje
        ?vag - vagoneta
        ?maq - maquina
        ?quant - numero
        ?lowerQuant - numero
        ?loc - localizacion
    )
    :precondition (and
        (danger ?eq) ;El equipaje debe ser peligroso
        (= ?loc OficinaInspeccion) ;Debemos estar en la oficina de inspección
        (at ?maq ?loc) ;La máquina también se debe encontrar en esa localización
        (follow ?vag ?maq) ;La vagoneta que contiene al equipaje debe seguir a la máquina
        (occupied ?vag ?eq) ;El equipaje debe estar cargado en la vagoneta
        (carried ?vag ?quant) ;Debe existir un número inferior al número de equipajes cargados en la vagoneta
        (next ?lowerQuant ?quant)
    )
    :effect (and
        (toanalyce ?eq) ;El equipaje se marcará como pendiente de ser analizado
        (not(occupied ?vag ?eq)) ;El equipaje ya no estará en la vagoneta
        (not(carried ?vag ?quant))  ;La cantidad de equipajes en la vagoneta pasará a ser el número inferior
        (carried ?vag ?lowerQuant)
    )
)
```

2.3.5. Conectar una vagoneta

En el caso de la acción de conectar una vagoneta es requerido que tanto la máquina como la vagoneta estén en la misma localización y que la parte del tren (máquina o vagoneta) a la que se va a conectar la nueva vagoneta debe formar parte del tren que dirije la máquina y también debe ser el último .^es^labón"de ese tren. No es necesario especificar que una vagoneta no debe estar ya conectada a la máquina dado que ya es necesario que se cumpla la condición de que esté .^es^perando.^a^ ser conectada en alguna localización, que no puede darse si la vagoneta estuviera

conectada a alguna máquina.
Como resultado, esta acción hará que la vagoneta deje de estar a la espera de ser conectada, que forme parte del tren que dirije la máquina y que pase a ser el último eslabón de ese tren.

Figura 8: Acción connect.

```
(:action connect
    :parameters (
        ?maq - maquina
        ?v1 - (either vagoneta maquina)
        ?v2 - vagoneta
        ?loc - localizacion
    )
    :precondition (and
        (waiting ?v2 ?loc) ;La vagoneta está suelta y a la espera de ser conectada en una localización
        (at ?maq ?loc) ;La máquina también se debe encontrar en esa localización
        (follow ?v1 ?maq) ;El último eslabón conectado sigue a la máquina
        (last ?maq ?v1) ;El último eslabón es realmente el último eslabón
    )
    :effect (and
        (not(waiting ?v2 ?loc)) ;La vagoneta deja de estar a la espera de ser conectada
        (follow ?v2 ?maq) ;La vagoneta pasará a aseguir también a la máquina
        (connected ?v2 ?v1) ;La vagoneta pasará a estar conectada detras el que era antes el último eslabón
        (not(last ?maq ?v1)) ;La vagoneta pasará a ser ahora el último eslabón del tren que dirige la máquina
        (last ?maq ?v2)
    )
)
```

2.3.6. Desconectar una vagoneta

Por otro lado, la acción de desconectar una vagoneta requiere que la vagoneta forme parte del tren que dirije la máquina, que sea el último eslabón de ese tren y que el número de equipajes cargados en esa vagoneta sea igual a cero. También es requerido que la vagoneta esté conectada a alguna otra parte del tren (máquina o vagoneta) y que ambas partes form parte a su vez del tren que dirije la máquina.
Esta acción hará que la vagoneta desconectada pase a estar esperando en la misma localización en la que se encuentra la máquina, que la vagoneta ya no forme parte del tren que dirije la máquina y que el último eslabón del tren sea aquel al que estaba conectado la vagoneta antes de ser desconectada.

Figura 9: Acción disconnect.

```
(:action disconnect
    :parameters (
        ?maq - maquina
        ?v1 - (either vagoneta maquina)
        ?v2 - vagoneta
        ?quant - numero
        ?loc - localizacion
    )
    :precondition (and
        (at ?maq ?loc) ;La máquina se encuentra en una cierta localización
        (follow ?v1 ?maq) ;El penúltimo eslabón sigue a la máquina
        (follow ?v2 ?maq) ;El último eslabón sigue a la máquina
        (connected ?v2 ?v1) ;Los eslabones último y penúltimo están conectados
        (last ?maq ?v2) ;El último eslabón es realmente el último eslabón
        (carried ?v2 zero) ;La cantidad de equipajes cargados en el último eslabón es cero
        (= ?quant zero)
    )
    :effect (and
        (waiting ?v2 ?loc) ;La vagoneta pasará a estar suelta y a la espera de ser conectada en esa localización
        (not(follow ?v2 ?maq)) ;La vagoneta dejará de seguir a la máquina
        (not(connected ?v2 ?v1)) ;La vagoneta dejará de estar conectada al que era el penúltimo eslabón
        (not(last ?maq ?v2)) ;El que era el penúltimo eslabón pasará a ser el último
        (last ?maq ?v1)
    )
)
```

2.3.7. Analizar un equipaje peligroso

Por último, la acción de analizar un equipaje peligroso simplemente requiere que un equipaje haya sido marcado como pendiente de analizar y que sea peligroso.
Como resultado, el equipaje dejará de ser peligroso, ya no estará pendiente de ser analizado y pasará a estar esperando a ser recogido por algún tren en la oficina de inspección.

Figura 10: Acción analyce.

```
(:action analyze
    :parameters (
        ?eq - equipaje
    )
    :precondition (and
        (toanalyce ?eq) ;El equiaje debe estar marcado como pendiente de ser analizado
        (danger ?eq) ;El equipaje debe ser peligroso
    )
    :effect (and
        (not(danger ?eq)) ;El equipaje deja de ser peligroso
        (notdanger ?eq) ;El equipaje pasa a ser seguro
        (not(toanalyce ?eq)) ;El equipaje deja de estar epndiente de ser analizado
        (waiting ?eq OficinaInspeccion) ;El equipaje pasa a estar disponible para ser cargado en la oficina
    )
)
```

2.4. Problema

Para la definición de la instancia del problema propuesto en el ejercicio es necesario especificar qué objetos van a participar, qué literales van a ser ciertos en el estado inicial y qué objetivos deberá cumplir el plan.
Los objetos declarados son las máquinas, vagonetas y equipajes existentes en el aeropuerto, así como las localizaciones y los números. Dado que las vagonetas sólo podrán cargar con 2 equipajes, únicamente podrán existir 3 números en el problema: zero, N1 y N2. Recordemos que el número 0 (zero) está definido como constante en el dominio.
Los literales que especificamos como ciertos en la definición del problema son:

- Relaciones de adyacencia entre cada par de localizaciones contiguas, en ambas direcciones.
- Posición y estado inicial de cada máquina. Cada máquina se sigue a sí misma y es el último eslabón del tren que dirige al no tener vagonetas conectadas.
- Posición de cada vagoneta/equipaje que está sin conectar/cargar en algún punto del aeropuerto.
- Destino de cada equipaje.
- Estado de un equipaje (seguro o peligroso).
- Cantidad de equipajes en cada vagoneta.
- Relación de orden entre los números declarados.

Por último indicamos que el plan debe conseguir que todos los equipajes se marquen como entregados en su destino como se muestra en la Figura 11.

Figura 11: Objetivos del problema.

```
(:goal (and
    (done E1)
    (done E2)
    (done E3)
    (done E4)
    (done E5)
    (done E6)
))
```

2.5. Análisis de resultados

El objetivo de este apartado es comparar los planes generados por los distintos planificadores sobre la instancia del problema y sobre variaciones de la misma.

Como se muestra en la Tabla 1, hemos generado planes para el problema propuesto en el ejercicio, para 2 variaciones del mismo y para 4 problemas más sencillos utilizados originalmente para debugging. En conjunto hemos probado diferentes números de máquinas, vagonetas, equipajes,

localizaciones y capacidad de las vagonetas.

Cuadro 1: Problemas utilizados en los experimentos.

Problema	**Localizaciones**	**Máquinas**	**Vagonetas**	**Equipajes**	**Números**	**Equip. cargados**
Original	10+1	2	5	6	2+1	No
Grande 1	10+1	2	5	6	10+1	No
Grande 2	10+1	2	5	10	2+1	No
Mini 1	2+1	1	1	1	2+1	No
Mini 2	2+1	1	2	3	2+1	Si
Mini 3	2+1	2	2	3	2+1	Si
Mini 4	2+1	2	2	3	2+1	No

MetricFF es un planificador secuencial, es decir, no tiene en cuenta el paralelismo de acciones ni se puede utilizar en dominios con acciones de duración variable. Por esta razón, para evaluar los planificadores en igualdad de condiciones, compararemos el tiempo hasta generar un plan y el número de acciones de cada plan independientemente de si están paralelizadas o no.

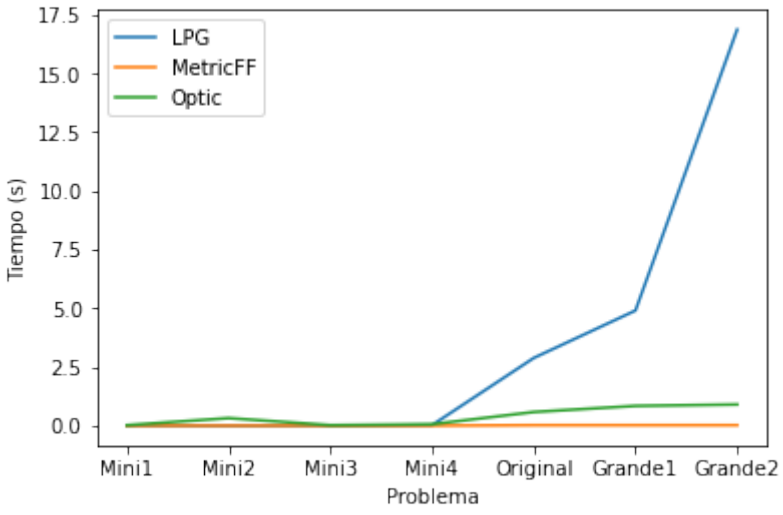
Los resultados obtenidos por los planificadores MetricFF, LPG y Optic para los distintos problemas problemas aparecen reflejados en la Tabla 2.

Cuadro 2: Resultados obtenidos en los experimentos.

		Mini 1	**Mini 2**	**Mini 3**	**Mini 4**	**Original**	**Grande 1**	**Grande 2**
LPG	Tiempo	∼0	∼0	∼0	0.02	2.88	4.9	16.86
	Acciones	9	11	11	23	43	46	63
MetricFF	Tiempo	∼0	∼0	∼0	∼0	0.02	0.02	0.02
	Acciones	9	11	11	16	44	40	64
Optic	Tiempo	0.02	0.32	0.02	0.06	0.58	0.84	0.9
	Acciones	9	11	11	24	53	49	76

De estos resultados podemos extraer que LPG es capaz de encontrar planes muy optimizados aunque el coste computacional aumenta mucho conforme aumentamos la talla del problema como se puede apreciar en la Figura 12.
En cambio, MetricFF puede alcanzar planes con un número de pasos similar en mucho menos tiempo por lo que parece ser la mejor opción como planificador en un dominio preposicional sin acciones temporales. Optic tarda algo más que MetricFF y por lo general no es capaz de encontrar planes mejores que los que proporciona MetricFF en cuanto a número de acciones.

Figura 12: Tiempo requerido para encontrar una solución.



3. Dominio temporal

Para el desarrollo del dominio temporal se proponen las siguientes restricciones:

- La acción de conectar/desconectar una vagoneta debe tener una duración fija.
- Debe haber máquinas tractores rápidas y lentas.
- Las localizaciones adyacentes deben tener una distancia que las separa.
- La acción de mover un tren tiene una duración proporcional a la velocidad de la máquina y a la distancia entre las localizaciones de origen y destino.
- Los equipajes pueden tener distinto peso.
- La acción de cargar/descargar un equipaje debe tener una duración proporcional al peso del equipaje.

A continuación se detallan los cambios realizados para poder cumplir con estas nuevas restricciones así como el análisis de los experimentos realizados.

3.1. Modificaciones sobre el dominio proposicional

3.1.1. Funciones

- (**distance ?l1** - localizacion **?l2** - localizacion)
  La función distancia nos indica la distancia entre dos localizaciones.
- (**speed ?maq** - maquina)
  La función velocidad nos indica la velocidad de una determinada máquina.
- (**capacity ?vag** - vagoneta)
  La función capacidad nos indica el número máximo de equipajes que se pueden cargar en una vagoneta.
- (**elements ?vag** - vagoneta)
  La función elementos nos indica el número de equipajes que se encuentran cargados en una vagoneta actualmente.
- (**weight ?eq** - equipaje)
  La función peso nos indica el peso de un equipaje.

Figura 13: Funciones añadidas.

```
(:functions
    (distance ?l1 - localizacion ?l2 - localizacion) ;Distancia entre localizaciones
    (speed ?maq - maquina) ;Velocidad de una máquina
    (capacity ?vag - vagoneta) ;Capacidad máxima de una vagoneta
    (elements ?vag - vagoneta) ;Cantidad actual de equipajes en una vagoneta
    (weight ?equip - equipaje) ;Peso de un equipaje (influye en el tiempo de carga/descarga)
)
```

3.1.2. Predicados

Ahora podemos prescindir del predicado (**next ?n1** - numero **?n2** - numero) dado que podemos usar números y compararlos directamente con el uso de operadores lógicos. También hemos eliminado el predicado (**carried ?vag** - vagoneta **?n** - numero) dado que ahora la cantidad de equipaje cargado en una vagoneta se almacena en la función (**elements ?vag** - vagoneta). No se ha añadido ningún nuevo predicado.

3.1.3. Acciones

- **Mover una máquina:**
  La duración de un movimiento se calcula como la división de la distancia recorrida entre la velocidad de la máquina.
  Como condiciones, la máquina debe estar en la localización de origen justo antes del inicio de la acción y las localizaciones inicio y destino deben seguir siendo adyacentes durante toda la acción.
  Como efectos, la máquina dejará de estar en la localización de origen al empezar la acción y estará en la localización de destino al terminar.

Figura 14: Acción temporal move.

```
(:durative-action move
    :parameters (
        ?maq - maquina
        ?orig - localizacion
        ?dest - localizacion
    )
    :duration (= ?duration (/ (distance ?orig ?dest) (speed ?maq))) ;En función de la distancia y la velocidad de la máquina
    :condition (and
        (at start (at ?maq ?orig))
        (over all (adyacent ?orig ?dest)) ;Las localizaciones deben seguir siendo adyacentes durante el proceso
    )
    :effect (and
        (at start (not (at ?maq ?orig))) ;La máquina deja de estar en el origen en cuanto empieza a moverse
        (at end (at ?maq ?dest)) ;Pasa a estar en el destino cuando el movimiento termina
    )
)
```

- **Cargar un equipaje:**
  La duración de esta acción se calcula como la multiplicación del peso del equipaje por un escalar. Si este escalar es uno, se tardarán tantas unidades de tiempo como unidades de peso tenga el equipaje.
  Justo antes de realizar la acción, el equipaje debe estar esperando a ser recogido en la localización y debe haber espacio en la vagoneta para cargarlo.
  Durante toda la acción la máquina debe permanecer en la localización y la vagoneta debe seguir siendo parte del tren.
  Al empezar la acción, el equipaje dejará de estar esperando y el número de equipajes dentro de la vagoneta incrementará en uno, reservando ese espacio en la vagoneta.
  Al terminar, el equipaje pasará a ocupar un espacio dentro de la vagoneta.

Figura 15: Acción temporal load.

```
(:durative-action load
    :parameters (
        ?equip - equipaje
        ?vag - vagoneta
        ?maq - maquina
        ?loc - localizacion
    )
    :duration (= ?duration (* 1.5 (weight ?equip))) ;La duración es relativa al peso del equipaje
    :condition (and
        (at start (waiting ?equip ?loc))
        (at start (> (capacity ?vag) (elements ?vag))) ;Debe quedar espacio para cargar el equipaje
        (over all (at ?maq ?loc)) ;La máquina debe permanecer quieta durante el proceso
        (over all (follow ?vag ?maq)) ;La vagoneta debe seguir unida al tren durante el proceso
    )
    :effect (and
        (at start (not(waiting ?equip ?loc))) ;El equipaje deja de estar disponible para ser usado por otras acciones durante el proceso
        (at start (increase (elements ?vag) 1)) ;El espacio que va a ocupar el equipaje en la vagoneta queda reservado al inicio del proceso
        (at end (occupied ?vag ?equip)) ;El equipaje puede ser descargado una vez está completamente cargado
    )
)
```

- **Descargar un equipaje seguro:**
  La duración de esta acción se calcula también como la multiplicación del peso del equipaje por un escalar.
  Justo antes de iniciar la acción, el equipaje debe estar cargado en la vagoneta y su destino debe ser aquel en el que se encuentra la máquina que sigue la vagoneta.
  Durante toda la acción la máquina debe permanecer en la misma localización, el equipaje debe seguir siendo seguro y la vagoneta debe seguir formando parte del tren que dirige la máquina.
  Justo al empezar la acción, el equipaje deja de tener un destino y deja de ocupar un espacio dentro de la vagoneta
  Al terminar la acción, el número de elementos dentro de la vagoneta decrementa en uno y el

equipaje se marca como entregado.

Figura 16: Acción temporal unload notdanger.

```
(:durative-action unload_notdanger
    :parameters (
        ?equip - equipaje
        ?vag - vagoneta
        ?maq - maquina
        ?loc - localizacion
    )
    :duration (= ?duration (* 1.5 (weight ?equip))) ;La duración es relativa al peso del equipaje
    :condition (and
        (at start (occupied ?vag ?equip))
        (at start (destination ?equip ?loc))
        (over all (at ?maq ?loc)) ;La máquina debe permanecer quieta durante el proceso
        (over all (follow ?vag ?maq)) ;La vagoneta debe seguir unida al tren durante el proceso
        (over all (notdanger ?equip)) ;El equipaje no puede pasar a ser peligroso durante el proceso
    )
    :effect (and
        (at start (not(destination ?equip ?loc))) ;El equipaje deja de estar disponible para ser cargado
        (at start (not(occupied ?vag ?equip))) ;Protegemos frente acciones que requieran que el equipaje esté dentro de la vagoneta
        (at end (decrease (elements ?vag) 1)) ;El espacio de la vagoneta que ocupaba el equipaje queda libre al completarse la descarga
        (at end (done ?equip))
    )
)
```

- **Descargar un equipaje peligroso:**
  La duración de esta acción se calcula también como la multiplicación del peso del equipaje por un escalar.
  Justo antes de iniciar la acción, el equipaje debe estar cargado en la vagoneta.
  Durante toda la acción la máquina debe permanecer en la oficina de inspección, la vagoneta debe seguir a la máquina y el equipaje debe seguir marcado como peligroso.
  Al empezar la acción, el equipaje deja de ocupar espacio dentro de la vagoneta, impidiendo que, durante el proceso, el equipaje pueda ser utilizado por otras acciones que requieran que esté ocupando espacio en la vagoneta.
  Al terminar, el número de elementos dentro de la vagoneta decrementa en uno y el equipaje se marca como pendiente de ser analizado.

Figura 17: Acción temporal unload danger.

```
(:durative-action unload_danger
    :parameters (
        ?equip - equipaje
        ?vag - vagoneta
        ?maq - maquina
    )
    :duration (= ?duration (* 1.5 (weight ?equip))) ;La duración es relativa al peso del equipaje
    :condition (and
        (at start (occupied ?vag ?equip))
        (over all (at ?maq OficinaInspeccion)) ;La máquina debe permanecer quieta durante el proceso
        (over all (follow ?vag ?maq))  ;La vagoneta debe seguir unida al tren durante el proceso
        (over all (danger ?equip)) ;El equipaje no puede pasar a ser seguro durante el proceso
    )
    :effect (and
        (at start (not(occupied ?vag ?equip))) ;Protegemos frente acciones que requieran que el equipaje esté dentro de la vagoneta
        (at end (decrease (elements ?vag) 1)) ;El espacio de la vagoneta que ocupaba el equipaje queda libre al completarse la descarga
        (at end (toanalyce ?equip)) ;El equipaje sólo se puede procesar cuando está totalmente descargado
    )
)
```

- **Conectar una vagoneta:**
  Como se indica en el ejercicio, la duración de esta acción es fija.
  Justo antes de iniciar la acción la vagoneta debe estar a la espera de ser conectada en la misma localización en la que se encuentra el tren al que se va a conectar. También antes de iniciar la acción, aquel eslabón (máquina o vagoneta) del tren al que se va a conectar la nueva vagoneta debe ser el último eslabón del tren, ya sea una vagoneta o la propia máquina en caso de no haber otras vagonetas conectadas a esa máquina.
  Durante toda la acción la máquina debe permanecer en esa localización y el eslabón al que se conectará la vagoneta debe seguir formando parte del tren.
  Justo al empezar la acción, la vagoneta dejará de estar a la espera de ser conectada y el que era el último eslabón del tren dejará de serlo. De este modo, mientras una vagoneta (v2) se conecta, el tren no tiene un "último eslabón" y se impide que el último eslabón original (v1) sea utilizado por otra acción como, por ejemplo, conectar otra vagoneta durante el proceso.
  Al terminar la acción, la vagoneta (v2) pasará a estar conectada al último eslabón original

(v1), seguirá también a la máquina y pasará a ser el nuevo último eslabón del tren.

Figura 18: Acción temporal connect.

```
(:durative-action connect
    :parameters (
        ?maq - maquina
        ?v1 - (either vagoneta maquina)
        ?v2 - vagoneta
        ?loc - localizacion
    )
    :duration (= ?duration 1) ;La duración es fija y corta
    :condition (and
        (at start (waiting ?v2 ?loc))
        (at start (last ?maq ?v1))
        (over all (at ?maq ?loc)) ;La máquina debe permanecer quieta durante el proceso
        (over all (follow ?v1 ?maq)) ;La penúltima vagoneta debe seguir conectada durante el proceso
    )
    :effect (and
        (at start (not(waiting ?v2 ?loc))) ;La vagoneta a conectar deja de estar disponible para ser conectada por otra máquina
        (at start (not(last ?maq ?v1))) ;La última vagoneta deja de ser el ultimo eslabón para evitar que otras vagonetas se puedan conectar
        (at end (connected ?v2 ?v1))
        (at end (follow ?v2 ?maq))
        (at end (last ?maq ?v2))
    )
)
```

- **Desconectar una vagoneta:**
  Como se indica en el ejercicio, la duración de esta acción es fija.
  Justo antes de iniciar la acción la vagoneta (v2) debe estar conectada detrás del penúltimo eslabón del tren (v1), siguiendo a la máquina.
  Durante toda la acción, la máquina debe permanecer en la misma localización, el penúltimo eslabón debe permanecer siguiendo a la máquina y no debe haber ningún equipaje dentro de la vagoneta que se va a desconectar.
  Al inicio de la acción, la vagoneta (v2) dejará de estar conectada al penúltimo eslabón (v1), dejará de seguir a la máquina y dejará de ser el último eslabón. De nuevo, durante la conexión, el tren no tendrá ningún "último eslabón".
  Al terminar la acción, el penúltimo eslabón (v1) pasará a ser el último. Además, la vagoneta quedará en la localización a la espera de ser conectada de nuevo.

Figura 19: Acción temporal disconnect.

```
(:durative-action disconnect
    :parameters (
        ?maq - maquina
        ?v1 - (either vagoneta maquina)
        ?v2 - vagoneta
        ?loc - localizacion
    )
    :duration (= ?duration 1) ;La duración es fija y corta
    :condition (and
        (at start (follow ?v2 ?maq))
        (at start (connected ?v2 ?v1))
        (at start (last ?maq ?v2))
        (over all (at ?maq ?loc))
        (over all (follow ?v1 ?maq)) ;La vagoneta de la que nos vamos a desconectar debe seguir a la máquina durante el proceso
        (over all (>= (elements ?v2) 0)) ;La vagoneta a desconectar debe estar vacía durante todo el proceso
        (over all (<= (elements ?v2) 0))
    )
    :effect (and
        (at start (not(follow ?v2 ?maq))) ;La vagoneta a desconectar deja de estar disponible para otras acciones que la requieran como parte del tren
        (at start (not(connected ?v2 ?v1))) ;La vagoneta a desconectar deja de estar disponible para otras acciones que la requieran como conectada
        (at start (not(last ?maq ?v2))) ;La vagoneta a desconectar deja de ser la última. Durante el proceso no hay "último eslabón"
        (at end (waiting ?v2 ?loc)) ;La vagoneta solo pasa a estar disponible para ser conectada al completarse el proceso
        (at end (last ?maq ?v1)) ;La penúltima vagoneta pasa a ser la última y queda disopnible para recibir conexiones al completarse el proceso
    )
)
```

- **Analizar un equipaje peligroso:**
  La duración del análisis de un equipaje peligroso es fija.
  Antes de iniciarse la acción, el equipaje debe estar marcado como pendiente de analizar y debe estar marcado como peligroso.
  Al iniciarse la acción, el equipaje deja de estar marcado como pendiente de analizar y deja de estar marcado como peligroso pero sin ser marcado como seguro. El equipaje estará en un estado indeterminado durante el proceso. De este modo, se impide que el equipaje sea utilizado por otra acción mientras está siendo analizado.
  Al terminar, el equipaje se marca como seguro y pasa a estar a la espera de ser recogido.

Figura 20: Acción temporal analyce.

```
(:durative-action analyze
    :parameters (
        ?equip - equipaje
    )
    :duration (= ?duration 3)
    :condition (and
        (at start (toanalyce ?equip))
        (at start (danger ?equip))
    )
    :effect (and
        (at start (not(toanalyce ?equip))) ;El eqipaje se está analizando y no está pendiente de ser analizado
        (at start (not(danger ?equip))) ;Protegemos al paquete de ser requerido por otras acciones como peligroso
        (at end (notdanger ?equip)) ;Solo pasa a ser seguro al completarse el análisis
        (at end (waiting ?equip OficinaInspeccion)) ;El equipaje pasa a estar disponible para ser cargado una vez terminado el proceso
    )
)
```

3.2. Modificaciones sobre el problema

A la hora de adaptar el problema al dominio temporal ha sido necesario inicializar las funciones descritas en el apartado Funciones: especificamos la distancia entre cada par de localizaciones adyacentes, la velocidad de cada máquina, la capacidad máxima de cada vagoneta, inicializamos la cantidad de equipajes de cada vagoneta a 0 y establecemos el peso de cada equipaje.
La métrica a minimizar será ahora el tiempo total.
Se han eliminado todas las referencias y hechos que utilizan predicados y tipos que han sido eliminados como (**next ?n1** - numero **?n2** - numero) y (**carried ?vag** - vagoneta **?n** - numero).

3.3. Análisis de resultados

En este apartado expondremos la experimentación realizada sobre el dominio temporal descrito anteriormente. En este caso, se ha probado el problema original de la tarea y también en otras variaciones de ese problema que influyan en la paralelización y duración final del plan. En concreto hemos probado a escalar por un determinado factor la distancia entre localizaciones, la velocidad de las máquinas y el peso de los equipajes.

Cuadro 3: Resultados obtenidos para el dominio temporal en LPG.

Distancias	Velocidades	Pesos	Duración
x1	x1	x1	906.000
x0.5	x1	x1	862.500
x1	x2	x1	464.500
x1	x1	x2	1347.000
x0.5	x2	x2	737.250

Cuadro 4: Resultados obtenidos para el dominio temporal en Optic.

Distancia	Velocidad	Equipajes	Duración
x1	x1	x1	637.011
x0.5	x1	x1	425.019
x1	x2	x1	425.019
x1	x1	x2	830.016
x0.5	x2	x2	770.638

Como podemos observar en las tablas de resultados, en general, Optic ha encontrado mejores planes que LPG. El escalado de variables refleja que, para el dominio implementado, el peso de los equipajes tiene un impacto notable en el tiempo del plan. Como era de esperar, escalar las distancias a la mitad equivale a escalar las velocidades de las máquinas al doble.

4. Dominio con recursos numéricos

En este ejercicio se nos pide adaptar el dominio temporal para dar soporte a nuevas restricciones que involucran el consumo eficiente de recursos que pueden ser renovables o no, a través del empleo de fluents en PDDL. Como se indica en el enunciado, este consumo debe ser inversamente proporcional al consumo de tiempo.

4.1. Recurso elegido

El recurso numérico que hemos planteado para este ejercicio es la gasolina que consumen las máquinas. Cada máquina podrá tener una velocidad distinta y un consumo por unidad de distancia distinto.
Se ha implementado una versión del dominio en la que la gasolina no es renovable, es decir, no se puede recargar el depósito de gasolina de una máquina. Después se ha añadido la posibilidad de recargar este depósito automáticamente cuando se agota independientemente de donde se encuentre la máquina.

4.2. Modificaciones sobre el dominio temporal

Para la versión no renovable fue necesario añadir nuevas funciones en el dominio así como su correspondiente inicialización en el problema.
Para la versión renovable fue necesario, además, crear una nueva acción para recargar el depósito de gasolina de una máquina.

4.2.1. Funciones

- (**max-fuel ?maq** - maquina)
  La función combustible máximo indica la capacidad máxima del depósito de una máquina.
- (**fuel ?maq** - maquina)
  La función combustible nos indica cuánta gasolina restante queda en el depósito de una máquina.
- (**burn ?maq** - maquina)
  La función consumo nos indica cuánta gasolina consume una máquina por unidad de distancia.
- (**total-fuel-used**)
  Esta función actúa como variable global e indica el consumo total de gasolina entre todas las máquinas del problema.
- (**refuel-duration**)
  Esta función también es una variable global e indica el tiempo requerido para cargar una unidad de combustible dentro de una máquina.

Figura 21: Funciones añadidas.

```
(max-fuel ?maq - maquina) 3👁 ;Máximo combustible que puede tener una máquina
(fuel ?maq - maquina) 3👁 1↘ 1= ;Combustible actual de una máquina
(burn ?maq - maquina) 3👁 ;Gasto de combustible de una máquina por unidad de distancia
(total-fuel-used) 1↗ ;Combustible total gastado
(refuel-duration) 1👁 ;Tiempo requerido para cargar una unidad de combustible dentro de una máquina
```

4.2.2. Acciones

- **Mover una máquina:**
  Antes de iniciar la acción, la máquina debe tener suficiente combustible como para llegar al destino.
  Al terminar, el combustible consumido se restará del depósito de la máquina y se sumará al total de combustible consumido.

Figura 22: Acción temporal move con recursos.

```
(:durative-action move
    :parameters (
        ?maq - maquina
        ?orig - localizacion
        ?dest - localizacion
    )
    :duration (= ?duration (/ (distance ?orig ?dest) (speed ?maq)))
    :condition (and
        (at start (at ?maq ?orig))
        (at start (>= (fuel ?maq) (* (distance ?orig ?dest) (burn ?maq)))) ;La máquina debe tener suficiente combustible como para llegar al destino
        (over all (adyacent ?orig ?dest))
    )
    :effect (and
        (at start (not (at ?maq ?orig)))
        (at end (at ?maq ?dest))
        (at end (decrease (fuel ?maq) (* (distance ?orig ?dest) (burn ?maq)))) ;El combustible consumido se resta al depósito de la máquina
        (at end (increase (total-fuel-used) (* (distance ?orig ?dest) (burn ?maq)))) ;El combustible consumido se suma al consumo total
    )
)
```

- **Recargar una máquina:**
  La duración de esta acción es proporcional a la cantidad de combustible que hay que verter en el depósito de la máquina.
  Justo antes de iniciar la acción, la máquina deberá haber agotado al menos la mitad de su combustible.
  Durante toda la acción la máquina debe permanecer en la misma localización sin moverse.
  Al terminar la acción, el depósito de combustible estará a su máxima capacidad.

Figura 23: Acción refuel.

```
(:durative-action refuel
    :parameters (
        ?maq - maquina
        ?loc - localizacion
    )
    :duration (= ?duration (* (- (max-fuel ?maq) (fuel ?maq)) (refuel-duration))) ;El tiempo empleado dependerá del combustible que haya que repostar
    :condition (and
        (at start (>= (/ (max-fuel ?maq) 2) (fuel ?maq))) ;Sólo puede recargar cuando tiene menos de la mitad del combustible
        (over all (at ?maq ?loc)) ;Mientras la máquina está repostando no se puede mover
    )
    :effect (and
        (at end (assign (fuel ?maq) (max-fuel ?maq))) ;La máquina estará a su máxima capacidad de combustible
    )
)
```

4.3. Modificaciones sobre el problema

Ahora es necesario introducir los valores numéricos iniciales de las funciones que hemos añadido. También deberemos inicializar las funciones globales: gasto total y tiempo requerido para repostar una unidad de combustible. Por último, especificaremos que la métrica a minimizar en el problema ya no es el tiempo empleado sino el total de gasolina consumido.

Figura 24: Minimización del consumo.

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

4.4. Análisis de resultados

En este apartado expondremos la experimentación realizada sobre el dominio temporal descrito anteriormente. En este caso, para los dominios renovable y no renovable, hemos realizado las siguientes comparativas:

- Consumo de combustible total variando el tamaño de los depósitos y la eficiencia de las máquinas.
- Tiempo y consumo total obtenido al minimizar tiempo o minimizar consumo total.

En cuanto a la primera comparativa, las siguientes tablas muestran los resultados obtenidos para cada planificador.

Cuadro 5: Efecto en el consumo del escalado de tamaño de los depósitos y la eficiencia de las máquinas (LPG).

Dominio	**Capacidad**	**Consumo**	**Consumo total**
No renovable	x1	x1	3840.00
No renovable	x2	x1	3400.00
No renovable	x1	x0.5	1420.00
No renovable	x2	x0.5	1780.00
Renovable	x1	x1	3220.00
Renovable	x2	x1	4180.00
Renovable	x1	x0.5	1320.00
Renovable	x2	x0.5	1540.00

Cuadro 6: Efecto en el consumo del escalado de tamaño de los depósitos y la eficiencia de las máquinas (Optic).

Dominio	**Capacidad**	**Consumo**	**Consumo total**
No renovable	x1	x1	2960.00
No renovable	x2	x1	2960.00
No renovable	x1	x0.5	1480.00
No renovable	x2	x0.5	1480.00
Renovable	x1	x1	2960.00
Renovable	x2	x1	2960.00
Renovable	x1	x0.5	1480.00
Renovable	x2	x0.5	1480.00

Como podemos apreciar en los resultados obtenidos, doblar la eficiencia de las máquinas reduce mucho más el gasto total que reducir a la mitad las distancias que las máquinas deben recorrer. De hecho, en Optic, aumentar la capacidad de los depósitos de las máquinas no tiene ningún efecto sobre el total de combustible consumido.

Respecto a la segunda comparativa, las siguientes tablas muestran el resultado obtenido para cada planificador.

Cuadro 7: Efecto de la minimización de tiempo o consumo (LPG).

Dominio	**Minimización**	**Duración**	**Consumo total**
No renovable	Consumo	756.000	3840.00
No renovable	Tiempo	728.500	Desconocido
Renovable	Consumo	746.000	3220.00
Renovable	Tiempo	831.500	Desconocido

Cuadro 8: Efecto de la minimización de tiempo o consumo (Optic).

Dominio	**Minimización**	**Duración**	**Consumo total**
No renovable	Consumo	670.011	2960.00
No renovable	Tiempo	572.019	Desconocido
Renovable	Consumo	670.011	2960.00
Renovable	Tiempo	670.011	Desconocido

De estos resultados podemos extraer que tanto en Optic como en LPG, minimizar la duración del plan parece resultar más fácil en el dominio no renovable que en el dominio renovable. Esto puede deberse a que no hemos dado tiempo suficiente a LPG y Optic a encontrar planes mejores al quedarnos con el segundo plan encontrado por LPG (-n 2) y el primero encontrado por Optic.

5. Desarrollo parcial de un árbol POP

Para el desarrollo del árbol POP se ha optado por trabajar con un dominio no instanciado, por tanto, la asignación de valor a las variables forma parte de los posibles flaws. Se aplica LIFO como

heurística de selección de flaws, de esta forma el último de los flaws detectados será el primero en ser resulto. Cada nodo tiene asignado un identificador de la forma $h.w$ siendo h su profundidad o nivel en el árbol y w la posición que ocupa en su correspondiente nivel (en caso de que se represente como $h.$ significa que solo un nodo existe en dicho nivel). El símbolo de la cruz roja indica la imposibilidad de seguir a través de ese camino. La elección del nodo a analizar en un nivel se realiza aleatoriamente.

Se han aplicado 10 iteraciones del algoritmo sobre el problema original, limitándonos a resolver el primer goal, por lo que el valor máximo que alcanzará h es 10 y w dependerá de las posibles formas de resolver el flaw seleccionado en su nivel.

A continuación, se detallan algunas observaciones importantes de cada uno de los niveles para una mejor comprensión del árbol.

Nivel 1

1. Se selecciona el goal **(done E1)** que solo puede ser resuelto por la acción **(unload_notdanger E1 ?v ?m ?n1 ?n2 ?l)**

Nivel 2

2. Se aplica **(unload_notdanger E1 ?v ?m ?n1 ?n2 ?l)** para la resolución del goal y se procede a resolver **(notdanger E1)** que puede lograrse aplicando **(analyze E1)** o directamente desde el nodo inicial.

Nivel 3

3.1 Se decide aplicar **(analyze E1)** e intentar proceder con el flaw **(danger E1)**. Notamos que es imposible resolver este flaw, por lo que se retorna al Nivel 2.

3.2 Creamos un enlace causal desde el nodo inicial hasta **(unload_notdanger E1 ?v ?m ?n1 ?n2 ?l)** y elegimos instanciar la variable **?l**.

Nivel 4

4.1 Asignamos **?l = P1** pero nos genera un conflicto con el enlace causal hacia el nodo inicial que no se puede resolver, por lo tanto retornamos al Nivel 3. Sucede de forma similar para 4.2 y 4.3.

4.4 Asignamos **?l = P4** y resolvemos **(destination E1 P4)** que solo puede obtenerse desde el estado inicial.

Nivel 5

5. Se resuelve **(destination E1 P4)** con el propio enlace causal hacia el nodo inicial y pasamos a instanciar **?m**.

Nivel 6

6.1 Instanciamos **?m = M1** y seleccionamos **(at M1 P4)** como próximo flaw que se genera de realizar la acción **(move M1 ?o P4)**.

Nivel 7

7. Tras aplicar **(move M1 ?o P4)** le daremos valor a **?o**.

Nivel 8

8.1 Intentamos realizar **?o = P1** pero tras seleccionar **(adyacent P1 P4)** notamos que no existe forma de resolverlo, por tanto volvemos al Nivel 7.

8.2 Asignamos **?o = P2** y tomamos **(adyacent P1 P4)** para ser analizado en el próximo nivel del árbol.

Nivel 9

9. Como solo se puede generar **(adyacent P1 P4)** desde el nodo inicial creamos el enlace causal pertinente y continuamos con el flaw **(at M1 P2)**.

Nivel 10

10. Notamos que solo se puede resolver **(at M1 P2)** con **(move M1 ?o P2)** y culminamos nuestro proceso tras su aplicación.

6. Graphplan

El grafo se presenta en un documento excel que se adjunta como parte del ejercicio. En el mismo se ven reflejadas todas las acciones requeridas por cada nivel, hasta alcanzar los objetivos propuestos, así como las precondiciones y efectos que causan. Por su parte, dado el volumen de acciones, se colocaron los índices de las precondiciones y efectos de todas las acciones de los 3 primeros niveles y en adelante solo los de aquellas que forman parte del plan relajado. La segunda hoja del documento, se muestran en que niveles fueron alcanzados los objetivos y los valores de las heurísticas h_sum, h_max y plan_relajado.

A continuación, procederemos a responder las preguntas planteadas.

1. Calcula el valor de las heurísticas h_sum y h_max para los dos objetivos definidos.

$h\_sum = 7 + 5 = 13$ dado que el primer objetivo fue alcanzado en el nivel 5 y el segundo en el nivel 7.

$h\_max = max(5, 7) = 7$ por alcanzarse el último objetivo en dicho nivel.

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

El plan relajado que se extrajo se observa en el documento a través de las acciones marcadas en color rojo. Se comienza por el último nivel y se aplican acciones que satisfagan los objetivos, a su vez, estas acciones poseen precondiciones que se convierten en nuevos objetivos. Se continua hasta satisfacer todos los objetivos y llegar al estado inicial del problema. Como no se toman en cuenta las exclusiones mutuas de las acciones y sus efectos negados, todas los efectos, una vez logrados, se mantienen para el resto de los niveles. Por este motivo es posible encontran caminos desde los objetivos hasta el estado inicial sin necesidad de hacer backtracking con una complejidad temporal polinómica.

3. ¿Cuál de las tres heurísticas calculadas (h_sum, h_max, plan_relajado) es la más informada para este problema? ¿Por qué?

La heurística más informada para nuestro problema consideramos que es h_max, pues gracias al del diseño del dominio se cuenta con muchas acciones que se podrían ejecutar de forma paralela para alcanzar ambos objetivos. Por su parte, h_sum y plan_relajado no explotan dicho beneficio.

7. Conclusiones

En este trabajo hemos analizado el dominio del aeropuerto y hemos extraído los predicados y acciones necesarios para la codificación de un dominio proposicional en PDDL poniendo en práctica los contenidos impartidos en las clases de teoría de la asignatura. Una vez codificado este dominio hemos generado planes para diferentes problemas comparando el rendimiento de cada planificador. A continuación hemos extendido nuestro dominio para soportar acciones temporales y recursos consumibles (renovables y no renovables), comparando cómo influye el escalado de distintas funciones en el tiempo total de los planes generados.

Se ha desarrollado el árbol POP aplicando de 10 iteraciones para observar el comportamiento de los planificadores parciales sobre nuestro problema. Por último hemos elaborado el grafo relajado reduciendo los objetivos iniciales y analizamos las principales heurísticas estudiadas.