

Razonamiento y planificación automática

Alejandro Cervantes

Resolución Actividad Laboratorio



Enunciado

Objetivos

Dominio: El mismo para toda la actividad (hecho por el estudiante)

Parte 1: Caso base

- Se indica la distribución concreta de contenedores, fábricas y depósito destino

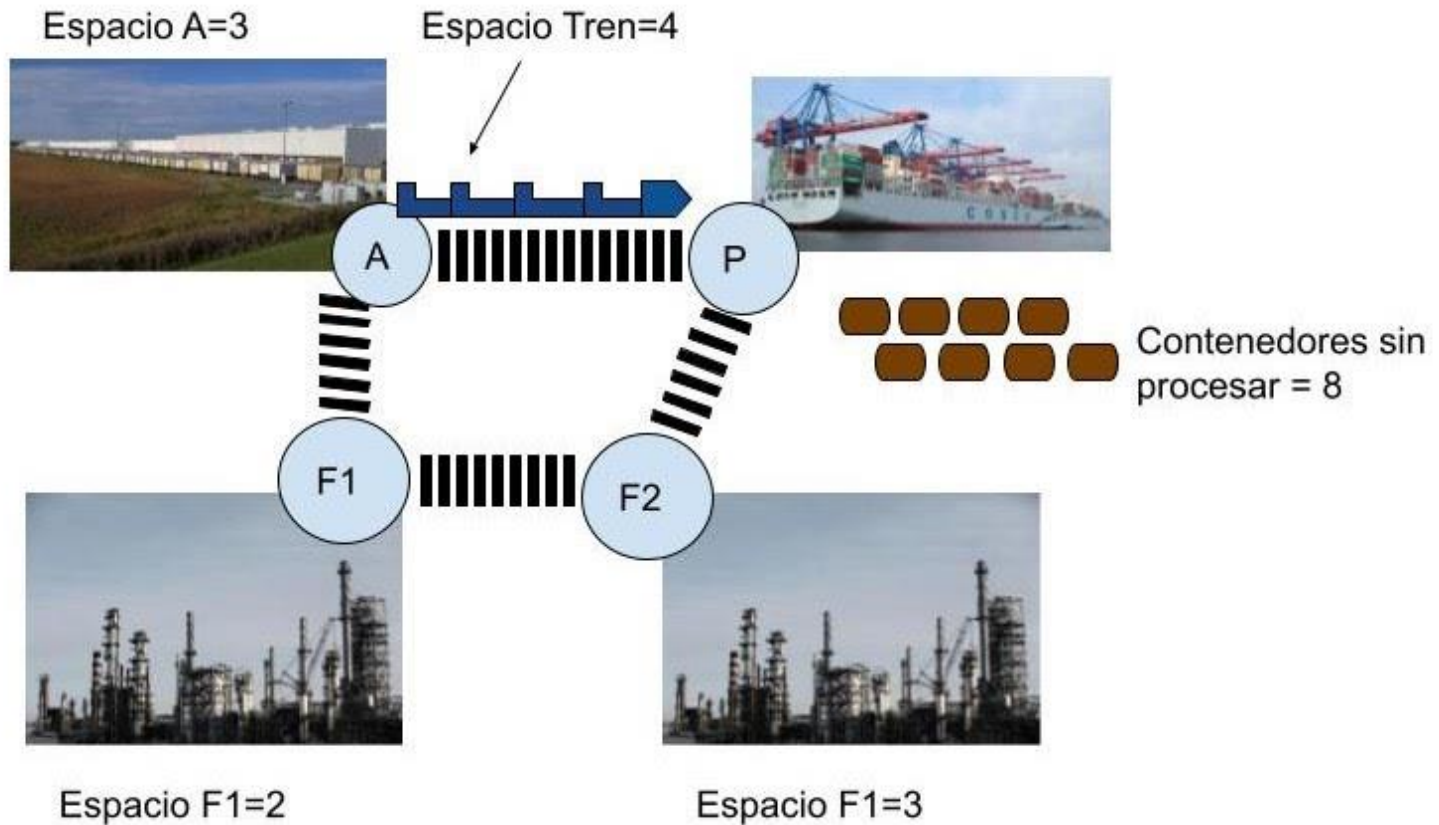
Parte 2: Modificaciones

- Se trata de hacer modificaciones interesantes del problema del caso base, manteniendo el dominio

Opciones: Varios planificadores (puntuación extra)

- Se puede instalar alguno de los planificadores mencionados y comparar resultados y posibilidades de cada uno. Aquí cabría usar variaciones del dominio (optimización, costes, números).

Dominio y caso base

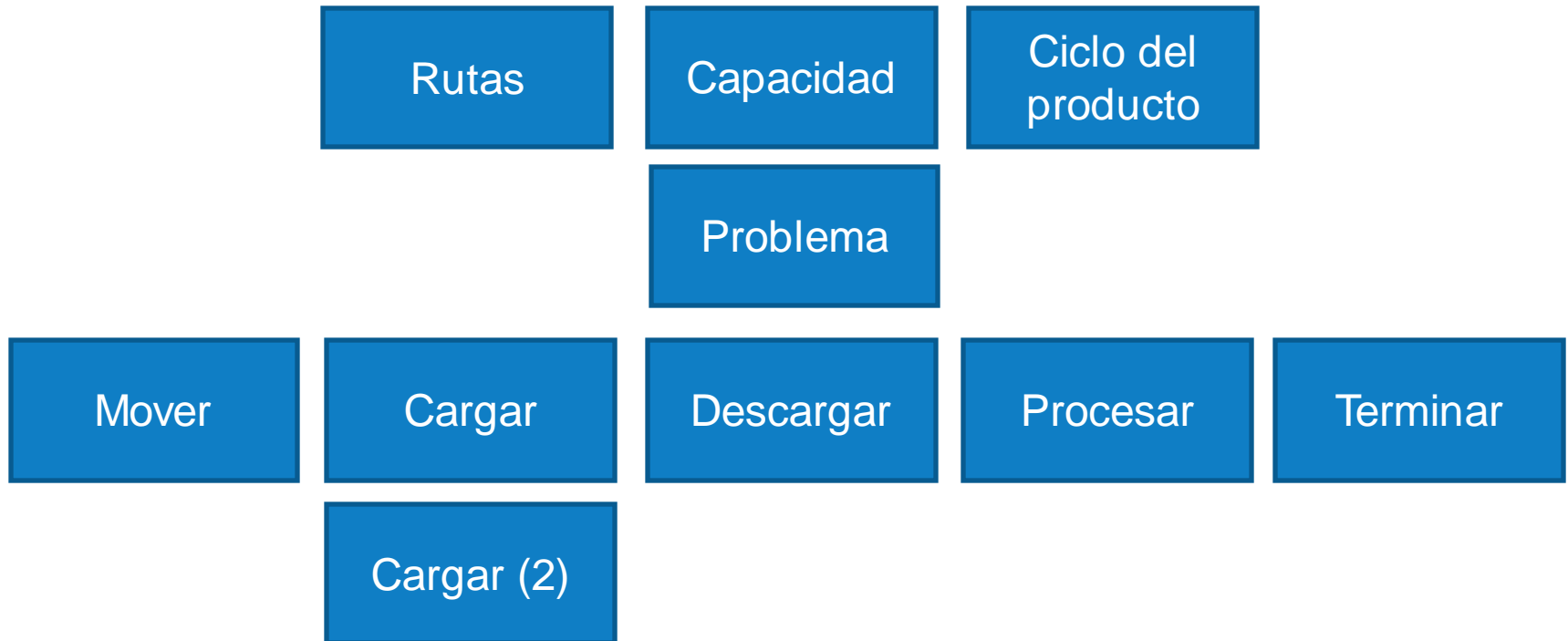


Operadores

- **Mover** a una localización **directamente conectada** sobre la vía (es decir, de A a P, o de F1 a A, etc.). **Puede moverse en direcciones opuestas en acciones sucesivas** (es decir, puede darse la vuelta ya que tiene máquina en ambos extremos).
- **Cargar** un contenedor en cualquier localización donde se encuentre, **siempre y cuando no se exceda el espacio disponible**.
- **Descargar** un contenedor en cualquier localización donde se encuentre, a excepción del puerto. Recuerde que al descargar no se puede exceder la capacidad de la localización.
- **Procesar** los contenedores que están en las fábricas (en esta acción el tren está parado)

Diseño

1. Tendremos diversos tipos de objeto, que podemos cualificar con tipos o con predicados (si sólo usamos STRIPS). Creamos problema.pddl
2. Hacemos una prueba para cada acción y así comprobamos la sintaxis más simple
3. Integramos todo



Movimiento

mover

Parámetros	Tren (opcional), origen, destino
Precondiciones	1. Tren está en origen 2. Conectado origen y destino + tipos (o bien los contenedores se moverán solos)
Efectos	Tren está en destino Tren deja de estar en el origen

Podríamos limitarnos a un solo tren, en ese caso no tendríamos el parámetro.

Al contrario, podríamos tener también distintos "tipos" de transporte (ver dominio Logistics) y entonces cualquiera de ellos podría mover, pero quizá lo hagan con operadores diferentes.

Movimiento

```
(define (domain trenes_movimiento)
  (:requirements :adl)
  (:predicates
    (en ?x ?y)
    (conectado ?x ?y)
    (movil ?x)
  )
  (:action mover
    :parameters (?tren ?origen ?destino)
    :precondition ( and
      (movil ?tren)
      (en ?origen ?tren)
      (conectado ?origen ?destino)
    )
    :effect ( and
      (not (en ?origen ?tren))
      (en ?destino ?tren)
    )
  )
)
```

```
(define (problem TRENES-MOVER)
  (:domain trenes_movimiento)
  (:objects tren1
             almacen fabrica1
             fabrica2 puerto
  )
  (:init
    (en puerto tren1)
    (movil tren1)
    (conectado puerto fabrica2)
    (conectado fabrica2 puerto)
    (conectado fabrica1 fabrica2)
    (conectado fabrica2 fabrica1)
    (conectado almacen fabrica1)
    (conectado fabrica1 almacen)
    (conectado almacen puerto)
    (conectado puerto almacen)
  )
  (:goal ( and (en fabrica1 tren1 )
  )
  )
  )
)
```


Carga (versión no generalizada)

cargar-en-fabrica-1	
Parámetros	Tren (opcional), contenedor, espacio-tren, espacio-fabrica-1
Precondiciones	<ol style="list-style-type: none">1. Tren y contenedor en fábrica 12. Hay espacio disponible en el tren3. El contenedor está en un espacio asociado a la fábrica 1
Efectos	<ul style="list-style-type: none">El contenedor deja de estar en la fábrica 1El espacio-tren pasa a estar ocupado por el contenedorEl espacio en la fábrica 1 gana una unidad



Añadimos la fábrica como parámetro

Medio bien: esto no sirve para extender a más fábricas.

Carga

cargar-en-puerto

Parámetros	Tren (opcional), lugar, contenedor, espacio-tren
Precondiciones	1. Tren está en mismo lugar que el contenedor 2. Hay sitio disponible en el tren 3. El lugar es un puerto (o no funciona con capacidad)
Efectos	El contenedor pasa a estar en el espacio del tren que estaba libre. El espacio-tren pasa a estar ocupado

cargar-en-otro

Parámetros	Tren (opcional), lugar , contenedor, espacio-tren, espacio-otro
Precondiciones	1. Tren está en el mismo lugar que el contenedor 2. Hay espacio disponible en el tren 3. El contenedor está en espacio-otro (asociado al lugar)
Efectos	El contenedor deja de estar en el lugar El espacio-tren pasa a estar ocupado por el contenedor El espacio-otro pasa a estar libre

Modelaremos los espacios con predicados "espacio" que podemos usar tanto para trenes como para fábricas, pero unos podrían ser vagones y otros "almacenes".

Los propios "espacios" están asociados a lugares

Para distinguir sitios hay que darles nombres (**objetos**) en el problema

Hacemos una regla particular para el puerto, que no tiene capacidad

Se podría también usar una única regla con **efectos condicionales** según el sitio

Carga en puerto

```
(define (domain trenes_carga)
  (:requirements :adl)
  (:predicates
    (en ?x ?y)
    (conectado ?x ?y)
    (movil ?x) (sinespacios ?x)
    (espacio ?objeto ?id ?carga)
    (cargable ?carga)
  )
  (:action cargar-en-puerto
    :parameters (?tren ?lugar ?contenedor ?id )
    :precondition ( and
      (en ?lugar ?tren)
      (en ?lugar ?contenedor) (sinespacios ?lugar)
      (cargable ?contenedor)      ; Lo que cargo se puede cargar
      (espacio ?tren ?id vacio) ; El tren tiene un espacio de id ?id vacio
    )
    :effect ( and
      (not (en ?lugar ?contenedor))
      (not (espacio ?tren ?id vacio))
      (espacio ?tren ?id ?contenedor)
    )
  )
)
```

Carga en puerto

```
(define (problem PROBLEMA-TRENES-CARGA-PUERTO)
  (:domain trenes_carga)
  (:objects tren1
    almacen fabrica1 fabrica2 puerto ; lugares
    c1 c2 c3 c4 c5 c6 c7 c8 ; contenedores
    v_1 v_2 v_3 v_4 ; lugares del tren
    vacio ; conveniencia, podria ser una constante
  )
  (:init
    (en puerto tren1)
    (movil tren1) (sinespacios puerto)
    (conectado puerto fabrica2) (conectado fabrica2 puerto)
    (conectado fabrica1 fabrica2) (conectado fabrica2 fabrica1)
    (conectado almacen fabrica1) (conectado fabrica1 almacen)
    (conectado almacen puerto) (conectado puerto almacen) ; Hasta aqui lo necesario para mover
    (en puerto c1) (en puerto c2) (en puerto c3) (en puerto c4)
    (en puerto c5) (en puerto c6) (en puerto c7) (en puerto c8) ; Ubicacion inicial de contenedores
    (espacio tren1 v_1 vacio) (espacio tren1 v_2 vacio)
    (espacio tren1 v_3 vacio) (espacio tren1 v_4 vacio) ; Espacio en el tren

    (cargable c1) (cargable c2) (cargable c3) (cargable c4)
    (cargable c5) (cargable c6) (cargable c7) (cargable c8) ; Evita cargar lugares en el tren
  )
  (:goal (and ; Ejemplo de objetivo, cargar totalmente el tren
    (not (espacio tren1 v_1 vacio))
    (not (espacio tren1 v_2 vacio))
    (not (espacio tren1 v_3 vacio))
    (not (espacio tren1 v_4 vacio))
  ))
  )
  )
  )
```

cargar-en-puerto tren1 puerto c6 v_1

cargar-en-puerto tren1 puerto c5 v_2

cargar-en-puerto tren1 puerto c4 v_3

cargar-en-puerto tren1 puerto c3 v_4

Carga en lugar con espacio de almacenamiento

```
(define (domain trenes_carga_2)
  (:requirements :adl)
  (:predicates
    (en ?x ?y) (conectado ?x ?y) (movil ?x)
    (sinespacios ?x) (espacio ?objeto ?id ?carga)
    (cargable ?carga)
  )
  (:constants
    vacio ; valor especial para espacios vacios
  )
  (:action cargar-en-otro
    :parameters (?tren ?lugar ?contenedor ?idvagon ?idotro )
    :precondition ( and
      (en ?lugar ?tren)
      (en ?lugar ?contenedor) ; Redundante, si el espacio esta en el lugar
      (cargable ?contenedor) ; Lo que cargo se puede cargar
      (espacio ?tren ?idvagon vacio) ; El tren tiene un espacio de id ?id vacio
      (espacio ?lugar ?idotro ?contenedor) ; El contenedor ya esta en un espacio
    )
    :effect ( and
      (not (en ?lugar ?contenedor))
      (not (espacio ?tren ?idvagon vacio))
      (not (espacio ?lugar ?idotro ?contenedor) )
      (espacio ?tren ?idvagon ?contenedor)
      (espacio ?lugar ?idotro vacio)
    )
  )
)
```

Carga en lugar con espacio de almacenamiento

```
(define (problem PROBLEMA-TRENES-CARGA-GENERAL)
  (:domain trenes_carga_2)
  (:objects tren1
    almacen fabrica1 fabrica2 puerto ; lugares
    c1 c2 c3 c4 c5 c6 c7 c8 ; contenedores
    v_1 v_2 v_3 v_4 ; lugares del tren
    f1_1 f1_2 ; lugares de la fabrica 1
    f2_1 f2_2 f2_3 ; lugares de la fabrica 2
    a_1 a_2 a_3 ; lugares del almacen
  )
  (:init
    (en fabrica1 tren1) (movil tren1) (sinespacios puerto)
    ; ... ojo, faltan cosas
    ; Ubicacion de los contenedores
    (en fabrica1 c1) (en fabrica1 c2) (en puerto c3) (en puerto c4)
    (en puerto c5) (en puerto c6) (en puerto c6) (en puerto c6)
    ; Espacios de las fabricas
    (espacio fabrica1 f1_1 c1) (espacio fabrica1 f1_2 c2)
    (espacio fabrica2 f2_1 vacio) (espacio fabrica2 f2_2 vacio) (espacio fabrica2 f2_3 vacio)
    ; Espacios del almacen
    (espacio almacen a_1 vacio) (espacio almacen a_2 vacio) (espacio almacen a_3 vacio)
    ; ... ojo, faltan cosas
  )
  (:goal (and ; Ejemplo de objetivo, vaciar una fabrica
    (espacio fabrica1 f1_1 vacio) (espacio fabrica1 f1_2 vacio)
  ))
)
```

cargar-en-otro tren1 fabrica1 c2 v_1 f1_2

cargar-en-otro tren1 fabrica1 c1 v_2 f1_1

Carga con efectos condicionales

```
(:action cargar
  :parameters (?tren ?lugar ?contenedor ?idvagon ?idotro )
  :precondition ( and
    (movil ?tren)
    (localizacion ?lugar)
    (en ?lugar ?tren)
    (espacio ?tren ?idvagon vacio) ; El tren tiene un espacio
    (cargable ?contenedor) ; Lo que cargo se puede cargar
    (en ?lugar ?contenedor)
  )
  :effect ( and
    (not (en ?lugar ?contenedor)) ; actualiza ubicación del contenedor
    (not (espacio ?tren ?idvagon vacio)) ; actualiza un vagón
    (espacio ?tren ?idvagon ?contenedor)
    (when
      (espacio ?lugar ?idotro ?contenedor) ; si hay contador
      ( and (not (espacio ?lugar ?idotro ?contenedor) )
        (espacio ?lugar ?idotro vacio) ; lo actualiza
      )
    )
  )
)
```

Descargar

descargar

Parámetros	Tren, lugar, contenedor, espacio-tren, espacio-otro
Precondiciones	<ol style="list-style-type: none">1. Tren lleva el contenedor (en un sitio-tren)2. Hay sitio disponible en el lugar (en espacio-otro)3. No estamos en el puerto (fallaría la 2 igualmente)
Efectos	<p>El contenedor pasa a estar en el espacio-otro que estaba vacío.</p> <p>El espacio-tren pasa a estar vacío</p>

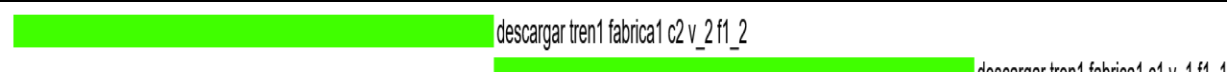
No se puede descargar en el puerto, pero basta para ello comprobar que hay espacios (no asignamos espacios al almacén)

Descargar no hace falta que deje el contenedor "en" la localización, ya que ese predicado no se usa para carga y descarga (lo deja "dentro" del espacio)

También se puede agrupar aquí el procesamiento y el envío (en más operadores)

Descarga en lugar con espacio de almacenamiento

```
(:action descargar
  :parameters (?tren ?lugar ?contenedor ?idvagon ?idotro )
  :precondition ( and
    (en ?lugar ?tren)
    (espacio ?tren ?idvagon ?contenedor) ; El tren tiene un
espacio de id ?id vacio
    (espacio ?lugar ?idotro vacio) ; El contenedor ya esta en un
espacio
    (cargable ?contenedor)      ; Lo que cargo se puede cargar
  )
  :effect ( and
    (not (espacio ?tren ?idvagon ?contenedor))
    (not (espacio ?lugar ?idotro vacio) )
    (espacio ?tren ?idvagon vacio)
    (espacio ?lugar ?idotro ?contenedor)
  )
)
```



Pero al añadir esta regla al resto, da un timeout (y error de memoria)

Descarga en lugar con espacio de almacenamiento

El problema de memoria quiere decir que el factor de ramificación es demasiado grande: las precondiciones se hacen verdad de muchas formas posibles.

```
(:action descargar
  :parameters (?tren ?lugar ?contenedor ?idvagon
?idotro )
  :precondition ( and
    (movil ?tren)
    (localizacion ?lugar)
    (en ?lugar ?tren)
    (espacio ?tren ?idvagon ?contenedor)
; El tren tiene un espacio de id ?id vacio
    (cargable ?contenedor) ; Lo que
cargo se puede cargar (no es vacio)
    (espacio ?lugar ?idotro vacio) ; El
contenedor ya esta en un espacio
  )
  :effect ( and
    (not (espacio ?tren ?idvagon ?contenedor))
    (not (espacio ?lugar ?idotro vacio) )
    (espacio ?tren ?idvagon vacio)
    (espacio ?lugar ?idotro ?contenedor)
  )
)
```

Al no usar tipos, se nos está olvidando en las precondiciones de las reglas filtrar los valores para los que se instancian. Por ejemplo "en" asume que hay dos lugares, pero no lo hemos especificado y por lo tanto se instancia para todos los pares de contenedores, espacios, etc.

Procesar y enviar

procesar

Parámetros	lugar, contenedor, sitio-otro
Precondiciones	<ol style="list-style-type: none">1. El contenedor está en un espacio de un lugar2. El contenedor no está procesado3. El lugar tiene capacidad de procesamiento
Efectos	El contenedor pasa a estar procesado (El tren no mueve)

enviar

Parámetros	lugar, contenedor, sitio-otro
Precondiciones	<ol style="list-style-type: none">1. El contenedor está en un espacio de un lugar2. El contenedor está procesado2. El lugar tiene capacidad de envío
Efectos	El contenedor pasa a estar enviado El contenedor se hace desaparecer para que no se siga moviendo (El tren no mueve)

Procesar y enviar

```
(:action procesar
  :parameters (?lugar ?contenedor ?idotro )
  :precondition ( and
    (puede_procesar ?lugar)
    (localizacion ?lugar)
    (espacio ?lugar ?idotro ?contenedor)
    (cargable ?contenedor)
    (not (procesado ?contenedor)) ; El contenedor no esta procesado
  )
  :effect ( and (procesado ?contenedor)
  )
)
```

```
(:action enviar
  :parameters (?lugar ?contenedor ?idotro )
  :precondition ( and
    (puede_enviar ?lugar)
    (localizacion ?lugar)
    (espacio ?lugar ?idotro ?contenedor) ; El contenedor ya esta en un espacio
    (cargable ?contenedor) ; Lo que cargo se puede cargar (no es vacio)
    (procesado ?contenedor) ; El contenedor no esta procesado
  )
  :effect ( and (enviado ?contenedor)
    (not (espacio ?lugar ?idotro ?contenedor) )
    (espacio ?lugar ?idotro vacio)
  )
)
```

Notas

- Es más cómodo utilizar **tipos** en lugar de predicados adicionales. Intentad aprender cómo usarlos.
- Si no se usan tipos, siempre "filtrar" para evitar que el planificador genere instancias de los operadores que son inválidas (mover un contenedor, cargar una fábrica, etc.)
- Si se generan demasiadas instancias se puede agotar la memoria, hay que añadir restricciones
- Si no se encuentra la solución puede que los operadores tengan restricciones incorrectas (o que supere el límite de tiempo impuesto)
- Se pueden añadir restricciones que proceden de nuestro conocimiento del dominio. Por ejemplo NO mover si hay algo por recoger. Puede facilitar la búsqueda (menos ramificación) y mejorar el plan. Puede haber casos en que esto impida encontrar el óptimo (es la duda sobre usar o no HTN)
- El planificador por omisión **no genera un plan óptimo**

Funcionamiento del planificador

- Un algoritmo óptimo obtendrá una solución de coste mínimo en caso de buscar el estado final como un único objetivo
- Sin embargo esto es irrealizable en la práctica (calculad la complejidad con vuestro dominio)
- Tampoco tenemos necesariamente una heurística que permita guiar una búsqueda informada
- En planificación, se buscan caminos que permitan resolver todas las metas abiertas aunque sea de forma independiente y luego fusionar planes
- Se puede calcular una heurística resolviendo una versión relajada del problema, que garantice que al menos se requiere un número de pasos

El planificador de Planning Domains

- Planning.domains utiliza el planificador **siw-then-bfs**. (<http://www.lapkt.org>) Tiene varios modos de funcionamiento y encadena varios algoritmos.

La salida del planificador indica que usa:

- **Primera opción:** usa un algoritmo llamado IW, como Breadth-First-Search pero descarta algunos nodos. La versión serializada (SIW) intenta encontrar cada subobjetivo de forma independiente, de modo que no se deshagan los subobjetivos anteriores. También se puede ejecutar sobre un problema relajado.
- **Si falla**, se combina con un algoritmo de búsqueda heurística avara (Best-first Search) cuya función de evaluación depende de la novedad del nodo y otros criterios (otros planificadores usan la primera pasada para calcular H de la segunda)
- Referencia: Christian Muise, Nir Lipovetzky, Miquel Ramirez (2015). [MAP-LAPKT: Omnipotent Multi-Agent Planning via Compilation to Classical Planning](#). *Proceedings of the Competition of Distributed and Multiagent Planners (CoDMAP)*.

Resultado

Planning service:

<http://solver.planning.domains/solve>

Domain: trenes_completo, Problem: PROBLEMA-TRENES

--- OK.

Match tree built with 616 nodes.

PDDL problem description loaded:

Domain: TRENES_COMPLETO

Problem: PROBLEMA-TRENES

#Actions: 616

#Fluents: 144

Landmarks found: 8

Starting search with IW (time budget is 60 secs)...

rel_plan size: 42

#RP_fluents 44

Caption

{#goals, #UNnachieved, #Achieved} ->

IW(max_w)

...

Total time: 0.468

Nodes generated during search: 12827

Nodes expanded during search: 1480

Plan found with cost: 71

BFS search completed

Efecto de los objetivos

- SIW busca subobjetivo a subobjetivo; si en el tiempo que se le da, encuentra el plan, el planificador da por terminada la búsqueda
- Si SIW no termina, pasamos a ejecutar BFS
- Si SIW no puede descomponer los objetivos, el comportamiento varía

```
(at package-1 Almacen)
(at package-2 Almacen)
(at package-3 Almacen)
(at package-4 Almacen)
(at package-5 Almacen)
(at package-6 Almacen)
(at package-7 Almacen)
(at package-8 Almacen)
(procesado package-1)
...
0.00100: (pick-up tren puerto package-1 capacity-3 capacity-4)
0.00200: (drive tren puerto f1)
0.00300: (procesar tren f1 package-1)
0.00400: (drive tren f1 f2)
0.00500: (drive tren f2 almacen)
0.00600: (drop tren almacen package-1 capacity-3 capacity-4)
0.00700: (drive tren almacen puerto)
0.00800: (pick-up tren puerto package-2 capacity-3 capacity-4)
0.00900: (drive tren puerto f1)
0.01000: (procesar tren f1 package-2)
0.01100: (drive tren f1 f2)
0.01200: (drive tren f2 almacen)
0.01300: (drop tren almacen package-2 capacity-3 capacity-4)
```

Efecto de los objetivos

- SIW busca subobjetivo a subobjetivo; si en el tiempo que se le da, encuentra el plan, el planificador da por terminada la búsqueda
- Si SIW no termina, pasamos a ejecutar BFS
- Si SIW no puede descomponer los objetivos, el comportamiento varía

```
(:goal (and (terminado) )  
)
```

BFS search completed

Plan found:

```
0.00100: (pick-up tren puerto package-1 capacity-3 capacity-4)  
0.00200: (pick-up tren puerto package-2 capacity-2 capacity-3)  
0.00300: (pick-up tren puerto package-3 capacity-1 capacity-2)  
0.00400: (drive tren puerto almacen)  
0.00500: (drop tren almacen package-3 capacity-1 capacity-2)  
0.00600: (drive tren almacen puerto)  
0.00700: (pick-up tren puerto package-4 capacity-1 capacity-2)  
0.00800: (drive tren puerto almacen)  
0.00900: (drive tren almacen f2)  
0.01000: (procesar tren f2 package-1)
```

Resultado

cargar-en-puerto tren1 puerto c8 v_4
cargar-en-puerto tren1 puerto c7 v_3
cargar-en-puerto tren1 puerto c6 v_2
mover tren1 puerto fabrica2
descargar tren1 fabrica2 c8 v_4 f2_3
mover tren1 fabrica2 puerto
cargar-en-puerto tren1 puerto c5 v_4
procesar fabrica2 c8 f2_3
mover tren1 puerto fabrica2
descargar tren1 fabrica2 c5 v_4 f2_1
procesar fabrica2 c5 f2_1
mover tren1 fabrica2 puerto
cargar-en-puerto tren1 puerto c4 v_4
mover tren1 puerto fabrica2
mover tren1 fabrica2 fabrica1
descargar tren1 fabrica1 c4 v_4 f1_2
procesar fabrica1 c4 f1_2
mover tren1 fabrica1 almacen
mover tren1 almacen puerto
cargar-en-puerto tren1 puerto c3 v_4
mover tren1 puerto almacen
mover tren1 almacen fabrica1
descargar tren1 fabrica1 c3 v_4 f1_1
procesar fabrica1 c3 f1_1
cargar-en-otro tren1 fabrica1 c3 v_1 f1_1
mover tren1 fabrica1 almacen
descargar tren1 almacen c3 v_1 a_3
enviar almacen c3 a_3
mover tren1 almacen puerto
cargar-en-puerto tren1 puerto c2 v_4
mover tren1 puerto almacen
mover tren1 almacen fabrica1
descargar tren1 fabrica1 c2 v_4 f1_1
cargar-en-otro tren1 fabrica1 c4 v_1 f1_2
procesar fabrica1 c2 f1_1
descargar tren1 fabrica1 c6 v_2 f1_2
procesar fabrica1 c6 f1_2
cargar-en-otro tren1 fabrica1 c2 v_2 f1_1
mover tren1 fabrica1 almacen
descargar tren1 almacen c4 v_1 a_2
enviar almacen c4 a_2
descargar tren1 almacen c2 v_2 a_3
mover tren1 almacen puerto
cargar-en-puerto tren1 puerto c1 v_4
mover tren1 puerto fabrica2
cargar-en-otro tren1 fabrica2 c5 v_2 f2_1
cargar-en-otro tren1 fabrica2 c8 v_1 f2_3
mover tren1 fabrica2 fabrica1
enviar almacen c2 a_3
mover tren1 fabrica1 almacen
descargar tren1 almacen c5 v_2 a_1
descargar tren1 almacen c8 v_1 a_2
mover tren1 almacen fabrica1
cargar-en-otro tren1 fabrica1 c6 v_2 f1_2
descargar tren1 fabrica1 c1 v_4 f1_1
procesar fabrica1 c1 f1_1
cargar-en-otro tren1 fabrica1 c1 v_1 f1_1
mover tren1 fabrica1 almacen
enviar almacen c5 a_1
enviar almacen c8 a_2
descargar tren1 almacen c1 v_1 a_3
enviar almacen c1 a_3
descargar tren1 almacen c6 v_2 a_2
enviar almacen c6 a_2
mover tren1 almacen fabrica1
descargar tren1 fabrica1 c7 v_3 f1_2
procesar fabrica1 c7 f1_2
cargar-en-otro tren1 fabrica1 c7 v_1 f1_2
mover tren1 fabrica1 almacen
descargar tren1 almacen c7 v_1 a_3
enviar almacen c7 a_3



www.unir.net