PLANIFICACIÓN AUTOMÁTICA

Práctica de laboratorio 2 - Planificación jerárquica con SHOP2

Grado en Ingeniería Informática
Universidad de Alcalá

Andrés Corbacho Sánchez
Gonzalo González Silverio
Javier Roma Rodríguez

Curso 2024-2025

CONTENIDO

1. EJERCICIO 1.1: LOGÍSTICA DE EMERGENCIAS EN SHOP2

1.1. MODELADO DEL DOMINIO

En este ejercicio, se ha diseñado un dominio de planificación para un dron encargado de realizar entregas de cajas a personas desde un depósito. Utilizando planificación jerárquica de tareas (HTN), se estructura el problema mediante operadores y métodos que permiten descomponer las tareas complejas en acciones más simples.

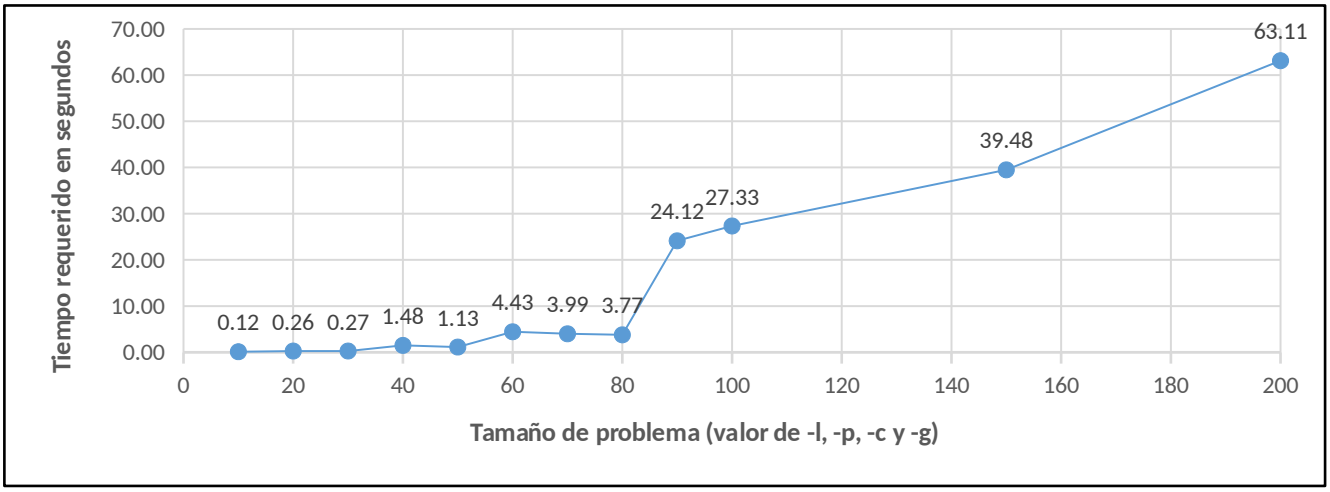
La construcción de los operadores se realizó a partir de las acciones del dominio PDDL del ejercicio 1.1 de la práctica 1, cuyas precondiciones y efectos fueron adaptados a la sintaxis de JSHOP2. Estos operadores son mover-dron, coger-brazo1, coger-brazo2, entregar-brazo1, y entregar-brazo2. El operador mover-dron permite que el dron se desplace de un lugar a otro. Los operadores coger-brazo1 y coger-brazo2 gestionan el proceso de recoger una caja con uno de los brazos del dron, asegurando que el brazo esté libre y que la caja se encuentre en el lugar adecuado. Los operadores entregar-brazo1 y entregar-brazo2 permiten entregar las cajas a las personas, eliminando la carga del brazo y actualizando las condiciones de la persona que recibe la caja.

Una vez definidos los operadores, se creó el método enviar-todo gestiona las entregas de las cajas en función de diferentes escenarios, siguiendo un enfoque recursivo con varios casos. En el primer caso, cuando el dron está en el depósito y hay dos cajas que deben ser entregadas a la misma persona, se utiliza un enfoque en el que el dron recoge ambas cajas con los dos brazos y las entrega de manera simultánea a la persona, para luego volver al depósito. En el segundo caso, cuando las dos cajas deben ser entregadas a dos personas distintas, el dron realiza las entregas en secuencia: primero lleva una caja a la primera persona, luego se desplaza al lugar de la segunda persona para entregar la otra caja. Finalmente, en el tercer caso, el dron gestiona la entrega de una sola caja a una persona. Si no hay más necesidades de entrega, el método finaliza en el caso base.

1.2. RESPUESTAS A LAS PREGUNTAS DEL EJERCICIO

1. Explica cómo crece el tiempo que tarda en generar una solución según el tamaño del problema, elaborando una gráfica.

Al inicio, los tiempos de ejecución son relativamente pequeños, oscilando entre 0,12 y 4,43 segundos para tamaños de problema que van desde 10 hasta 80. Sin embargo, a partir del tamaño 90, el tiempo de resolución comienza a crecer de manera notable, alcanzando los 24,12 segundos. A medida que el tamaño del problema sigue aumentando, el tiempo de ejecución continúa incrementándose, y para el tamaño 200, supera los 60 segundos, llegando a 63,11 segundos. Esto refleja cómo la complejidad del problema aumenta de manera significativa a medida que los parámetros de entrada (-l, -p, -c, -g) se incrementan, lo que lleva a un aumento exponencial en el tiempo requerido para encontrar una solución. A continuación, se muestra una gráfica que resume los resultados obtenidos:



2. Compara los resultados anteriores con los del planificador que mejor rendimiento mostró en el ejercicio 1.3 de la práctica 1.

Comparando los resultados obtenidos en el ejercicio actual con los del planificador LPG-TD, que mostró el mejor rendimiento en el ejercicio 1.3 de la práctica 1, se puede observar que en el ejercicio actual, utilizando JSHOP2, se obtienen mejores resultados en cuanto a la capacidad para manejar problemas de mayor tamaño. Mientras que LPG-TD supera el minuto de ejecución con un problema de tamaño 127 (donde el tamaño del problema se refiere a los valores de los parámetros -l, -p, -c y -g), en el ejercicio actual, los tiempos de ejecución con JSHOP2 apenas superan los 60 segundos en el tamaño 200, alcanzando 63 segundos. Esto indica que JSHOP2 es más eficiente en la gestión de problemas de mayor tamaño dentro del tiempo límite, ya que no se ve afectado por el crecimiento del tamaño de manera tan pronunciada como LPG-TD, que tiene una capacidad más limitada para problemas de gran complejidad.

3. Explica los resultados y reflexiona sobre las ventajas e inconvenientes de la planificación jerárquica respecto a la planificación clásica.

En el ejercicio actual, la planificación jerárquica mediante JSHOP2 demuestra ventajas claras en cuanto a la capacidad de resolver problemas de mayor tamaño en comparación con la planificación clásica. Esto se refleja en los resultados obtenidos, donde JSHOP2, que utiliza planificación jerárquica, es capaz de resolver problemas más grandes en menos tiempo que los planificadores clásicos. Por ejemplo, JSHOP2 supera el tamaño de 127, que es el límite para LPG-TD, y resuelve problemas de hasta 200 en solo 63 segundos. Esta diferencia se debe a que, en la planificación jerárquica, las tareas complejas se descomponen en subtareas más pequeñas, lo que permite gestionar de manera más eficiente los problemas grandes. Otro aspecto clave es el conocimiento experto incorporado en los métodos de refinamiento de tareas, que facilita una planificación más estructurada y optimizada, especialmente cuando se manejan grandes volúmenes de tareas o se requiere un control preciso sobre la secuenciación de acciones.

Por otro lado, la planificación clásica, aunque más flexible y generalista, tiene dificultades para manejar problemas de mayor escala debido a su enfoque más directo y menos estructurado. Aunque los planificadores clásicos como LPG-TD pueden manejar problemas complejos, no alcanzan la capacidad de JSHOP2 en términos de tamaño y velocidad. Esto sugiere que la planificación jerárquica ofrece un rendimiento superior cuando se trata de problemas más complejos, gracias a la capacidad de descomponer y paralelizar tareas de manera eficiente.

Sin embargo, el principal inconveniente de la planificación jerárquica es la dependencia del conocimiento experto. Si bien JSHOP2 es muy eficiente en problemas bien estructurados y con tareas claramente descomponibles, esta misma dependencia puede hacer que la planificación jerárquica sea menos flexible ante problemas nuevos o no previstos sin modificar el conocimiento de las tareas. En contraste, la planificación clásica, al no requerir este tipo de conocimiento específico, es más adaptable, aunque con un rendimiento limitado en problemas de mayor tamaño.

2. EJERCICIO 1.2: LOGÍSTICA DE EMERGENCIAS AVANZADA EN SHOP2

2.1 MODELADO DEL DOMINIO

A nivel conceptual el dominio reproduce la logística de emergencias como un sistema de inventarios numéricos gobernado por tareas jerárquicas. Los únicos objetos discretos que siguen existiendo son los contenedores de movimiento—drones y transportadores—y las localizaciones; todo lo demás (personas, cajas individuales) se abstrae mediante fluents que cuentan cuántas cajas quedan por entregar, cuántas hay almacenadas o qué capacidad libre conserva un vehículo.

Tipos y hechos estáticos
Se declaran tres clases de objetos: DRON, TRANSPORTADOR y LOCALIZACION; además, una lista cerrada de tipos de contenido (CONTENIDO) que en los problemas de prueba son solo “comida” y “medicina”. Los predicados dron-en y

transportador-en sitúan cada vehículo, mientras que limite-transportador define la capacidad máxima de cada transportador.

Fluents de inventario y necesidad
La cantidad de cajas se codifica con tres predicados numéricos:

- loc-necesita(loc, ct, n) indica cuántas cajas de cierto contenido todavía faltan en la localización.
- loc-total-necesita(loc, n) guarda el total pendiente—facilita saber si un destino ya está servido.
- loc-tiene(loc, ct, n) refleja el stock realmente presente (el depósito parte con todo el inventario, los destinos con cero).

Para los transportadores existen dos fluents complementarios: transportador-tiene(t, ct, n) y capacidad-transportador(t, libre). El primero cuenta las cajas ya cargadas, el segundo cuántas posiciones libres quedan. El par se actualiza en espejo cada vez que se carga o descarga para evitar sobrepasar el limite-transportador.

Operadores primitivos
El conjunto de acciones cubre: mover un dron solo, mover dron + transportador, cargar y descargar cajas entre depósito, transportador y destino, y por último entregar la caja en el punto de ayuda. Cada operador posee precondiciones numéricas coherentes (por ejemplo, solo se puede «dejar en transportador» si la capacidad libre es mayor que cero) y actualiza los contadores correspondientes. La acción de desplazamiento con transportador computa su coste como 50+capacidad/2 por tramo, cumpliendo el criterio del enunciado de 50 unidades base más un recargo proporcional.

Métodos HTN
La tarea raíz enviar-todo busca, de forma recursiva, la primera localización que todavía necesite cajas y delega la entrega en realizar-entrega. Ese método tiene dos ramas exclusivas: si el destino requiere exactamente una caja se opta por un dron en solitario; si necesita más de una se activa el flujo con transportador. Para los envíos masivos, una cadena de sub-métodos se encarga de:

1. cargar-transportador mientras queden unidades por transportar y espacio libre;
2. volar a destino con mover-transportador;
3. descargar iterativamente con coger-de-transportador y entregar-contenido;
4. regresar al depósito para reanudar el ciclo si aún hay más destinos pendientes.

Esa jerarquía garantiza lo requerido: un vehículo visita un solo destino por viaje, regresa siempre al depósito y nunca se embarca un transportador cuando basta con un dron.

Política de decisiones implícita
El método elige el primer transportador disponible cuyo espacio libre permita atender la demanda, pero la función de coste hace que, cuando haya varios vehículos, SHOP2 tienda a preferir el de menor capacidad (y por tanto menor recargo) hasta que ese no resulte ya suficiente; de esta manera el plan suele ser cercano al óptimo en costes sin necesidad de heurísticas adicionales.

2.2 DEMOSTRACIÓN DEL FUNCIONAMIENTO

50 cajas en localización, con 10 de limite en transportador. Coste del plan: 550.0. Time Used = 0.019

El listado refleja cómo SHOP2 satisface una petición de 50 cajas con un único transportador de capacidad 10. Primero vemos que el dron carga, una a una, diez cajas de comida en el transportador; cuando está lleno, despega hacia *loc1* (mover-transportador deposito loc1). En destino descarga las diez cajas (coger-de-transportador … loc1 + entregar-contenido … loc1) y regresa al depósito. Ese patrón—cargar 10, volar, descargar, volver—se repite cinco veces: tres tandas sólo con comida y dos tandas mixtas que incluyen la medicina. El coste total impreso (550) lo confirma: cada trayecto cuesta 55 puntos (50 fijos + 10/2 de recargo), así que un ida-y-vuelta son 110 y cinco viajes suman exactamente 550.

Aunque las fases de carga aparecen muy largas—SHOP2 muestra cada "coger/dejar" por separado—en realidad no se sobrepasa la capacidad porque cada grupo de diez pares precede a un único vuelo. Tras la quinta vuelta el fluents loc-total-necesita llega a cero y el método raíz enviar-todo termina. El resultado es un plan que cumple todas las reglas del dominio: nunca se transportan más de diez cajas a la vez, el dron sólo visita un destino por viaje y siempre regresa al depósito antes de iniciar la siguiente salida.

51 cajas en localización, con 10 de limite en transportador. Coste del plan: 650.0. Time Used = 0.017

En esta ejecución SHOP2 afronta un destino que requiere 51 cajas disponiendo solo de un transportador con límite 10. El planificador genera cinco ciclos idénticos en los que el dron carga diez cajas en el transportador, vuela a *loc1*, las descarga una a una y vuelve al depósito. Cada ida-y-vuelta con ese vehículo cuesta 50 + 10/2 = 55 por tramo, es decir 110 por viaje completo; cinco viajes para 50 cajas suponen 550 unidades de coste.

Una vez entregadas esas 50 cajas, todavía falta una unidad. Como el método HTN establece que si el destino necesita solo una caja se prescinde del transportador, el dron recoge la última medicina, vuela en solitario a *loc1*, la entrega y regresa vacío al depósito. Ese trayecto sin transportador cuesta 50 por tramo, así que el viaje de ida y vuelta añade 100 unidades. La suma de los cinco viajes con transportador (550) más el viaje directo del dron (100) da exactamente el coste total 650 que reporta el plan. El resultado ilustra cómo el dominio hace convivir ambas políticas: utiliza el transportador cuando compensa y recurre al dron solo para el remanente, respetando capacidad, costes y la regla de "un destino por viaje".

50 cajas en localización, con 40 de limite en transportador. Coste del plan: 280.0. Time Used = 0.02

El dominio identifica que el destino necesita 50 cajas y que el único transportador disponible admite hasta 40. Con esa información el método HTN decide fraccionar la entrega en dos viajes de ida y vuelta.
En el primer bloque del plan el dron carga 40 unidades (todas las que caben) en el transportador 1; a continuación despega a *loc1*, descarga las 40 cajas una a una y regresa vacío al depósito. El coste de un trayecto con este vehículo es 50 + 40/2 = 70; por tanto el ida-y-vuelta consume 140 puntos.

Al volver todavía quedan 10 cajas pendientes. El dron vuelve a emplear el transportador—aunque sobre mucho espacio libre—porque el método "rama transportador" se activa siempre que la necesidad sea mayor que una caja. Tras cargar esas 10 unidades repite el ciclo: vuelo al destino, descarga y retorno. Ese segundo viaje añade otros 140 puntos. La suma de los dos viajes explica el coste total 280 registrado por SHOP2.

Las largas secuencias "coger/dejar" que anteceden a cada vuelo sólo reflejan la carga caja-a-caja; funcionalmente el plan respeta las reglas del dominio: nunca supera la capacidad declarada, atiende un único destino por viaje y vuelve al depósito antes de iniciar el siguiente desplazamiento.

50 cajas en localización con transportadores con límites 10,20,30 . Coste del plan: 360.0. Time Used = 0.018

El plan muestra cómo el dominio reparte las 50 cajas que necesita *loc1* usando los tres transportadores disponibles (10, 20 y 30 plazas) en orden de menor a mayor capacidad para reducir el recargo por viaje. En la primera fase el dron llena el transportador 1 con 10 cajas de comida, vuela al destino, las descarga y regresa; cada tramo con ese vehículo cuesta 50 + 10/2 = 55 de modo que la ida-y-vuelta suma 110 puntos.

Aún quedan 40 cajas. El método elige ahora el transportador 2 (límite 20). El dron carga 20 unidades de medicina, completa el mismo ciclo depósito-destino-depósito y paga 50 + 20/2 = 60 por tramo, es decir 120 puntos adicionales.

Por último, restan 20 cajas mixtas. Como el transportador de 20 ya está ocupado, el plan emplea el transportador 3 (límite 30). Después de cargar las 20 cajas se realiza el tercer vuelo; con un recargo de 50 + 30/2 = 65 por tramo la ida-y-vuelta añade 130 puntos.

La suma de los tres desplazamientos—110 + 120 + 130—da el coste total 360 que reporta SHOP2. El resultado confirma la política codificada en el dominio: se envía siempre un solo destino por viaje, nunca se rebasa la capacidad declarada y se intenta usar primero el transportador más pequeño que aún permite atender la carga pendiente para minimizar el coste global.

50 cajas en localización con transportadores con límites 20,50,100 . Coste del plan: 270.0. Time Used = 0.019

El plan de 270 puntos muestra cómo el método elige siempre el transportador más pequeño que basta para la carga pendiente, reduciendo el recargo variable (capacidad ÷ 2) que se añade al coste fijo de 50 por vuelo. Primero llena hasta el tope el transportador 1 (capacidad 20) con 20 cajas de comida; el dron vuela depósito → loc 1, ejecuta 20 entregas y regresa vacío, lo que cuesta $(50 + 20/2) \times 2 = 120$puntos.

Al quedar 30 cajas de medicina, resultaría menos eficiente hacer dos viajes más con el transportador 1, así que se carga todo de una vez en el transportador 2 (capacidad 50). Tras 30 recogidas y su traslado a loc 1, el dron efectúa 30 entregas de medicina, descarga además tres cajas de comida sobrantes y vuelve al depósito; este segundo trayecto cuesta $(50 + 50/2) \times 2 = 150$ puntos. La suma 120 + 150 concuerda con el coste total de 270 que informa SHOP2, confirmando que el modelo respeta las capacidades, agrupa entregas y minimiza vuelos.

20,50,100 cajas en localizaciones con transportadores con límites 20,50,100 . Coste del plan: 470.0. Time Used = 0.039

El trazado corresponde al escenario con tres destinos (loc 1, loc 2 y loc 3) que precisan 20, 50 y 100 cajas respectivamente, y tres transportadores cuyas capacidades coinciden con esos valores (20, 50 y 100). El plan opera en serie: primero carga 20 cajas de medicina en el transportador 1, vuela al loc 1, descarga una a una y regresa; a continuación, llena el transportador 2 con las 50 cajas de comida destinadas al loc 2 y repite el ciclo; finalmente embarca en el transportador 3 una mezcla de 100 cajas (comida + medicina) para el loc 3, donde también se efectúan las entregas individuales antes de volver al depósito. Cada bloque de acciones coger-contenido / dejar-en-transportador materializa la actualización numérica de cajas en depósito y vehículo; la acción mover-transportador realiza un "viaje" (ida y vuelta) y las parejas coger-de-transportador / entregar-contenido satisfacen la necesidad de la localización destino sin violar la restricción de que un dron sólo visita un destino por viaje.

El coste total de 470 refleja la política de costes: a cada vuelo se le suma un recargo igual a la mitad de la capacidad del transportador además del fijo de 50, y se contabiliza tanto la ida como la vuelta. Así, el trayecto con la plataforma de 20 cajas cuesta $(50 + 20/2) \times 2 = 120$; el de 50 cajas, $(50 + 50/2) \times 2 = 150$; y el de 100 cajas, $(50 + 100/2) \times 2 = 200$. La suma 120+150+200 produce exactamente los 470 que SHOP2 reporta, evidenciando que el método selecciona el transportador mínimo que cubre cada demanda y evita fragmentar los envíos, lo cual hubiera incrementado el número de vuelos y, por tanto, el coste.

50,100 cajas en localizaciones con transportador de límite 150 . Coste del plan: 500.0. Time Used = 0.028

En este trazado el dominio afronta el caso en el que dos destinos (loc 1 y loc 2) requieren, respectivamente, 50 cajas mezcladas de comida y medicina, y sólo se dispone de un único transportador cuya capacidad total es 150 cajas. El dron recurre a la misma plataforma para atender los dos pedidos de forma secuencial: primero ejecuta un ciclo de carga en el depósito donde traslada, fluently, todas las cajas de comida y luego todas las de medicina al transportador 1; cuando la bodega numérica alcanza las 50 cajas se activa mover-transportador, el vehículo vuela al loc 1 y allí se repiten parejas coger-de-transportador / entregar-contenido hasta dejar la necesidad a cero antes de regresar al depósito.

Una vez de vuelta, el método reabastece la misma plataforma con otra tanda de comida y medicina destinada al loc 2 y repite exactamente el mismo patrón de entrega. Como el coste base de cada "viaje" es 50 y se añade un recargo igual a la mitad de la capacidad del transportador, cada ida-y-vuelta con la plataforma de 150 cajas cuesta $(50 + 150/2) \times 2 = 250$. Al realizarse dos viajes independientes (uno por destino) el total asciende a 500, valor que SHOP2 reporta al final del plan. La ejecución demuestra que los métodos seleccionan el único transportador disponible, satisfacen la restricción de "un destino por viaje" y evitan subdividir cargas, cumpliendo con las reglas de coste del dominio avanzado.

3. EJERCICIO 1.3: OPTIMIZACIÓN DEL PLANIFICADOR

Para la optimización del planificador hemos llevado a cabo las tareas propuestas en el enunciado, para ello ha sido necesario modificar varias partes de los metodos anteriores, a continuación explicamos como se ha llevado a cabo cada mejora:

```
(:method (enviar-todo)
    (:sort-by ?n >
    (and
        (loc-total-necesita ?l ?n)
        (call > ?n 0)))
    (
    (realizar-entrega ?l ?n)

    (enviar-todo)
    )
    ()
    ()
)
```

1. **Generar planes en los que se atienda primero a los lugares que mayor necesidad total de recursos tienen.**

    - **(:sort-by ?n > ...)**

      Esta cláusula fuerza al planificador a generar primero el subplan para el lugar ?l que tenga el valor ?n de loc-total-necesita más alto.

    - **Filtrado call > ?n 0**

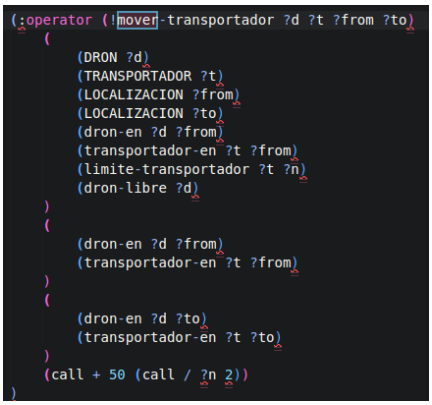
      Sólo se consideran los lugares que aún tienen necesidad pendiente (n > 0).

    De este modo, en lugar de ir destino por destino secuencialmente, el plan recorre recursivamente la lista ordenada de mayores a menores necesidades, garantizando que los más urgentes se atiendan primero.

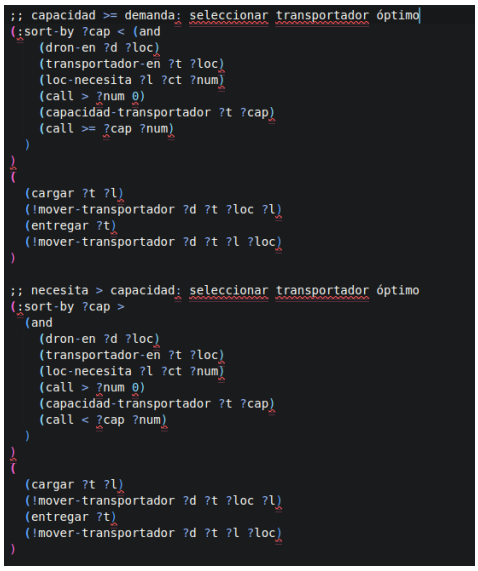
2. **Generar planes que hagan un uso eficiente de los transportadores eligiendo el transportador a enviar según su capacidad de carga y el número de cajas que se necesitan en el destino.**

   Para usar bien los transportadores, es necesario que el planificador no los trate todos igual, sino que prefiera aquel cuya capacidad encaje mejor con el número de cajas que hay que mover.

   - **Coste dinámico del operador de movimiento**



     - Al incorporar al coste la fórmula 50 + (límite/2), el planner tenderá a elegir el transportador que minimice ese coste.
     - Un transportador con capacidad más ajustada (ni muy grande ni muy pequeño respecto a la demanda) resultará más barato de mover.
     - Metodo realizar-entrega modificado, modificamos el método para que:
       - **Si hay algún transportador cuya capacidad (`?cap`) sea al menos igual a la demanda**, se elige el de menor capacidad posible (para no sobredimensionar).
       - **Si ninguno cubre toda la demanda, se elige el transportador de mayor capacidad disponible** (para aprovechar al máximo lo que exista).



3. **Comparación de Resultados Post-Optimización**

- **50 cajas en localización, con 10 de limite en transportador**.
  - Pre → Coste del plan: 550.0. Time Used = 0.019
  - Post Optimización → Coste del plan: 550.0. Time Used = 0.017
- **51 cajas en localización, con 10 de limite en transportador.**
  - Pre-Optimización → Coste del plan: 650.0. Time Used = 0.017
  - Post-Optimización → Coste del plan: 650.0. Time Used = 0.017
- **50 cajas en localización, con 40 de limite en transportador.**
  - Pre-Optimización → Coste del plan: 280.0. Time Used = 0.02
  - Post-Optimización → Coste del plan: 280.0. Time Used = 0.02

Hasta aquí los resultados iban a ser iguales, debido a q no se aplica ninguna de las reglas ya que solo hay un transportador y una localización

- **50 cajas en localización con transportadores con límites 10,20,30 .**
  - Pre-Optimización → **C**oste del plan: 360.0. Time Used = 0.018
  - Post-Optimización → Coste del plan: 250.0. Time Used = 0.017
- **50 cajas en localización con transportadores con límites 20,50,100 .**
  - Pre-Optimización → Coste del plan: 270.0. Time Used = 0.019
  - Post-Optimización → Coste del plan: 150.0. Time Used = 0.015

En los 2 problemas de arriba podemos observar como la segunda regla se aplica correctamente, distribuyendo de manera mejor las cantidades entre transportes

Primer problema: usa primero el transportador 3 que es el que mas capacidad tiene y despues usa el transportador 2 q es el q mejor se ajusta la capacidad

```
(!entregar-contenido dron1 medicina loc1)
(!coger-de-transportador dron1 transportador3 medicina loc1)
(!entregar-contenido dron1 medicina loc1)
(!mover-transportador dron1 transportador3 loc1 deposito)
(!coger-contenido dron1 medicina deposito)
(!dejar-en-transportador dron1 transportador2 medicina deposito)
```

Segundo problema usa el transportador 2 que es el que mejor se ajusta a la capacidad

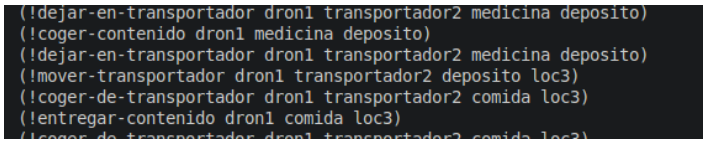
```
(!coger-contenido dron1 medicina deposito)
(!dejar-en-transportador dron1 transportador2 medicina depo
(!mover-transportador dron1 transportador2 deposito loc1)
(!coger-de-transportador dron1 transportador2 comida loc1)
(!entregar-contenido dron1 comida loc1)
```

- **50,100 cajas en localizaciones con transportador de límite 150 . Coste del plan: 500.0. Time Used = 0.028**
    - Pre-Optimización → Coste del plan: 500.0. Time Used = 0.028
    - Post-Optimización → Coste del plan: 500.0. Time Used = 0.025

  En esta el coste se mantiene ya que solo hay un transportador, pero el orden de reparto cambia entregando primero a loc2 que es el que tiene mayor necesidad

- **20,50,100 cajas en localizaciones con transportadores con límites 20,50,100 .**
    - Pre-Optimización → Coste del plan: 470.0. Time Used = 0.039
    - Post-Optimización → Coste del plan: 780.0. Time Used = 0.03

  El Coste aumenta ya que el orden de prioridades cambia, y obliga al planificador a seguir un orden de entrega, entregando primero a los mas necesitados, aun asi la optimización del reparto de materiales hace que el coste total sea 780 en vez de 980 que es solo con la priorizacion de localizaciones.



4. CONCLUSIONES

En esta práctica se ha modelado con éxito un sistema de logística de emergencias utilizando planificación jerárquica (HTN) en SHOP2. A través de los tres ejercicios, se demostró la eficacia del enfoque jerárquico tanto en escenarios discretos como numéricos, permitiendo una gestión eficiente y estructurada de entregas con drones y transportadores. Los resultados evidencian que SHOP2 supera a planificadores clásicos como LPG-TD en problemas de mayor escala, gracias a la descomposición de tareas y el uso de conocimiento experto. Sin embargo, esta eficiencia depende de una buena estructuración del dominio, lo que puede limitar su flexibilidad ante escenarios nuevos.