

*Resolución de un problema
de planificación logística
mediante técnicas de
Inteligencia Artificial*

PLANIFICACIÓN LOGÍSTICA

Realizado por Carlos Gálvez

Actividad grupal: Planificación logística

Contenido

<i>Introducción</i>	2
2. Modelado del problema con PDDL	2
2.1 Descripción del escenario	2
2.2 Estructura del dominio.....	2
2.3. Estructura del problema	2
3. Resolución del problema mediante búsqueda A*	4
3.1. Función de evaluación.....	4
3.2. Desarrollo de la búsqueda y estados generados	4
3.4 Análisis de la heurística	6
4. Grafo de planificación (GraphPlan)	7
5. Conclusión	9

Introducción

La planificación automática es una de las áreas fundamentales dentro de la Inteligencia Artificial, especialmente en entornos donde es necesario generar secuencias de acciones que permitan alcanzar un objetivo definido desde un estado inicial.

El desarrollo de la actividad se ha estructurado en tres fases principales. En primer lugar, se ha modelado el problema utilizando el lenguaje PDDL definiendo tanto el dominio de acciones como el escenario concreto del problema. A continuación, se ha resuelto el problema mediante el algoritmo de búsqueda A*, lo que ha permitido identificar un plan óptimo. Finalmente, se ha construido un grafo de planificación parcial (GraphPlan) y se han analizado las relaciones de exclusión mutua (mutex) entre acciones y proposiciones, fundamentales para la comprensión de las limitaciones estructurales en entornos de planificación.

2. Modelado del problema con PDDL

2.1 Descripción del escenario

El problema se enmarca en una situación logística donde un camión debe recoger un paquete en una ciudad de origen y transportarlo a una ciudad de destino. El entorno consta de tres ciudades, con conexiones bidireccionales entre ciudad1 y ciudad2, y entre ciudad2 y ciudad3. No existe conexión directa entre ciudad1 y ciudad3, lo que obliga al camión a pasar por ciudad2. El paquete comienza en ciudad1 y debe terminar en ciudad3, siendo necesario cargarlo y descargarlo explícitamente para completar el objetivo.

2.2 Estructura del dominio

Para representar este problema se ha definido un dominio PDDL que incluye:

- Tipos: ciudades, camiones y paquetes.
- Predicados: ubicación del camión, ubicación del paquete, si el paquete está en el camión, y conexiones entre ciudades.
- Acciones: **mover**, **cargar** y **descargar**, cada una con sus precondiciones y efectos.

2.3. Estructura del problema

En el fichero de problema se han especificado:

- Los objetos involucrados: un camión, un paquete y tres ciudades.
- El estado inicial: el camión está en ciudad2 y el paquete en ciudad1.
- El objetivo: que el paquete se encuentre en ciudad3.

Dominio PDDL

```
(define (domain transporte)
  (:requirements :strips :typing)

  (:types
    ciudad
    camion
    paquete
  )

  (:predicates
    (camion-en ?c - camion ?ci1 - ciudad)
    (paquete-en ?p - paquete ?ci1 - ciudad)
    (paquete-en-camion ?p - paquete ?c - camion)
    (conexion ?ci1 - ciudad ?ci2 - ciudad)
  )

  (:action mover
    :parameters (?c - camion ?from - ciudad ?to - ciudad)
    :precondition (and
      (camion-en ?c ?from)
      (conexion ?from ?to)
    )
    :effect (and
      (not (camion-en ?c ?from))
      (camion-en ?c ?to)
    )
  )

  (:action cargar
    :parameters (?p - paquete ?c - camion ?ci1 - ciudad)
    :precondition (and
      (camion-en ?c ?ci1)
      (paquete-en ?p ?ci1)
    )
    :effect (and
      (not (paquete-en ?p ?ci1))
      (paquete-en-camion ?p ?c)
    )
  )
)
```

Problema PDDL

```
(define (problem problema-transporte)
  (:domain transporte)

  (:objects
    ciudad1 ciudad2 ciudad3 - ciudad
    camion1 - camion
    paquetel - paquete
  )

  (:init
    (camion-en camion1 ciudad2)
    (paquete-en paquetel ciudad1)
    (conexion ciudad1 ciudad2)
    (conexion ciudad2 ciudad1)
    (conexion ciudad2 ciudad3)
    (conexion ciudad3 ciudad2)
  )

  (:goal
    (and
      (paquete-en paquetel ciudad3)
    )
  )
)
```

3. Resolución del problema mediante búsqueda A*

Este algoritmo se basa en la evaluación de los posibles estados del problema mediante una función de coste $f(n)$ que combina el coste real acumulado hasta el estado $g(n)$ con una estimación heurística del coste restante $h(n)$.

3.1. Función de evaluación

La función de evaluación utilizada es la siguiente:

$$f(n) = g(n) + h(n)$$

Donde:

- $g(n)$ representa el número de acciones desde el estado inicial hasta el estado actual.
- $h(n)$ es la heurística proporcionada en el enunciado:
 - $h(n) = 3 \rightarrow$ si el paquete no está en el camión ni en la ciudad destino.
 - $h(n) = 1 \rightarrow$ si el paquete está dentro del camión.
 - $h(n) = 0 \rightarrow$ si el paquete está en la ciudad destino.

3.2. Desarrollo de la búsqueda y estados generados

Estado inicial y objetivo

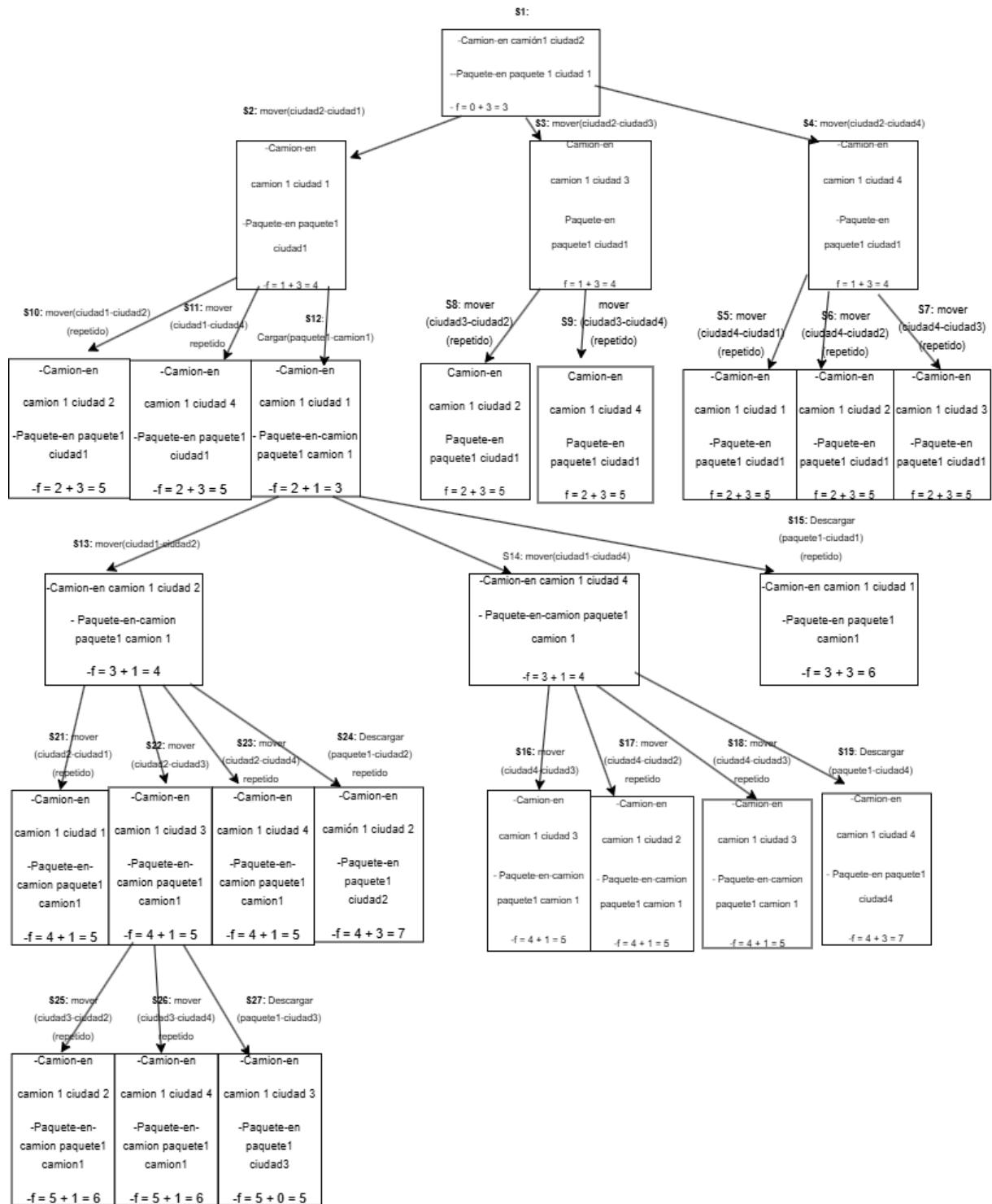
- **Estado inicial:** el camión (camion1) se encuentra en la ciudad2, y el paquete (paquete1) está en la ciudad1.
- **Objetivo:** conseguir que el paquete esté en la ciudad3. Para ello, el camión debe ir hasta la ciudad1, cargar el paquete, trasladarse hasta la ciudad3 y descargarlo.

Cabe destacar que no existe conexión directa entre la ciudad1 y la ciudad3, por lo que será necesario pasar obligatoriamente por la ciudad2.

Árbol de búsqueda y estados generados

***nota: en cada estado se calcula directamente la función de evaluación por simplicidad no se dice que es, se evalua según el orden $f(s) = g(s) + h(s)$**

A continuación, se describen las iteraciones que realiza el algoritmo A* para encontrar la solución:



*nota: Los estados se numeran en orden de cuando se expanden y los estados repetidos solo se nombran una vez y luego se suprinen.

1º iteración:

- Lista abierta = S1-3

2º iteración

- Lista Cerrada = S1-3
- Lista abierta = S4-4, S3-4, S2-4

3º Iteración:

- Lista Cerrada = S1-3, S4-4
- Lista abierta = S3-4, S2-4, S7-5(repetido), S6-5(repetido), S5-5(repetido)

4º Iteración:

- Lista Cerrada = S1-3, S4-4, S3-4
- Lista abierta = S2-4, S9-5(repetido), S8-5(repetido)

5º Iteración:

- Lista Cerrada = S1-3, S4-4, S3-4, S2-4
- Lista abierta = S12-3, S11-5, S10-5

6º Iteración:

- Lista Cerrada = S1-3, S4-4, S3-4, S2-4, S12-3
- Lista abierta = S14-4, S13-4, S11-5, S10-5, S15-6(repetido)

7º Iteración:

- Lista Cerrada = S1-3, S4-4, S3-4, S2-4, S12-3, S14-4
- Lista abierta = S13-4, S18-5(repetido), S17-5(repetido), S16-5, S11-5, S10-5

8º Iteración:

- Lista Cerrada = S1-3, S4-4, S3-4, S2-4, S12-3, S14-4, S13-4
- Lista abierta = S23-5(repetido), S22-5, S21-5(repetido), S16-5, S11-5, S10-5, S24-7(repetido)

9º Iteración:

- Lista Cerrada = S1-3, S4-4, S3-4, S2-4, S12-3, S14-4, S13-4, S22-5
- Lista abierta = S27-5, S22-5, S16-5, S11-5, S10-5, S26-6, S25-6
- Como S27 que es el estado meta esta al principio de lista abierta se coloca en la siguiente iteración en lista cerrada y ya se ha acabado.

10º iteración:

- Lista Cerrada = S1-3, S4-4, S3-4, S2-4, S12-3, S14-4, S13-4, S22-5, S27-5

3.4 Análisis de la heurística

- **¿Es admisible la heurística?**

Sí, es admisible. Una heurística se considera admisible cuando nunca sobreestima el coste real para alcanzar el objetivo. En este caso, Se puede ver en el grafo que no hay ningún estado que sobreestime el coste a la meta.

- **¿Es consistente la heurística?**

También es consistente. Una heurística es consistente si el valor estimado no disminuye más de lo que se avanza en el camino, es decir, si cumple que

$$h(n) \leq c(n, n') + h(n')$$

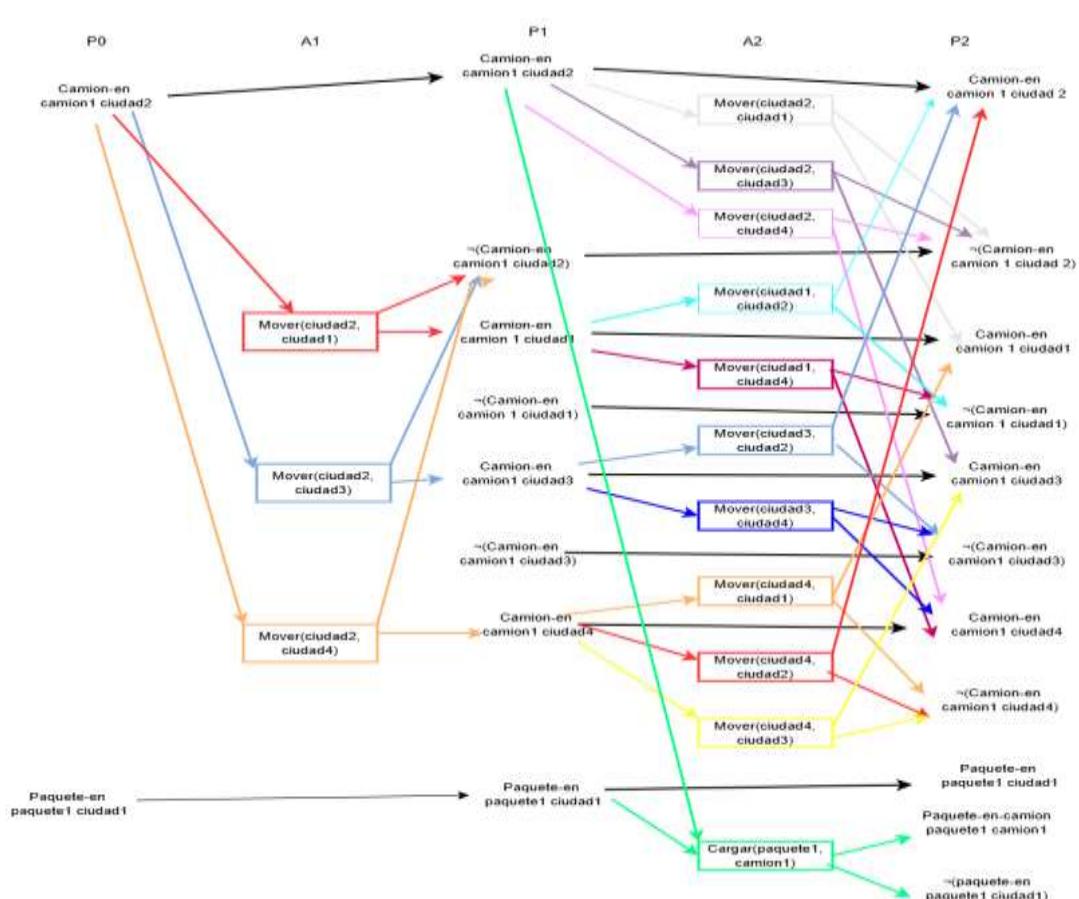
para cualquier transición entre estados. En este caso, se puede ver en el grafo que no hay ninguna función de evaluación de un estado hijo que sea mayor que el estado padre

4. Grafo de planificación (GraphPlan)

4.1 Estructura del grafo

Para analizar la evolución del problema a lo largo del tiempo, se ha construido un grafo de planificación parcial que alterna niveles de proposiciones (P) y acciones (A).

Se han desarrollado los niveles P[0], A[1], P[1], A[2], P[2], con el siguiente significado:



***nota:** Por simplicidad y que quede limpio el graphplan en cada acción de moverse con el camión no hemos relacionado la precondición de que no este en ninguna posición que no sea la de origen por ejemplo la acción mover(ciudad1,ciudad2) debería relacionarse con $\neg(\text{camion-en camion1 ciudad2})$ y $\neg(\text{camion-en camion1 ciudad3})$ y $\neg(\text{camion-en camion1 ciudad4})$ pero a la hora de indicar ejemplos de mutex por necesidades competitivas se utilizarán

4.2 Relaciones mutex

Durante la construcción del grafo, se han identificado **relaciones de exclusión mutua (mutex)** entre acciones y proposiciones. A continuación, se describen dos ejemplos de cada uno de los tipos requeridos:

◊ Efectos inconsistentes

1º En A2 las acciones Mover(ciudad2, ciudad1) y Mover(ciudad1, ciudad2) producen efectos inconsistentes ya que, el efecto de una acción niega el efecto de la otra. La acción Mover(ciudad2, ciudad1) produce como efecto $\neg(\text{Camion-en camion1 ciudad2})$ y la acción Mover(ciudad1, ciudad2) produce la negación del efecto de la otra que es Camion-en camion1 ciudad2)

2º En A2 las acciones Mover(ciudad2, ciudad3) y Mover(ciudad1, ciudad2) producen efectos inconsistentes ya que, el efecto de una acción niega el efecto de la otra. La acción Mover(ciudad2, ciudad3) produce como efecto $\neg(\text{Camion-en camion1 ciudad2})$ y la acción Mover(ciudad1, ciudad2) produce la negación del efecto de la otra que es $\neg(\text{Camion-en camion1 ciudad2})$

◊ Interferencia

1º Mover(ciudad2, ciudad1) es mutex con mover(ciudad2, ciudad3) por interferencias en el nivel A0 porque Mover(ciudad2, ciudad1) tiene como precondición la proposición (Camion-en camion1 ciudad2)y la acción mover(ciudad2, ciudad3) produce un efecto que niega esta proposición.

2º Mover(ciudad2, ciudad3) es mutex con mover(ciudad2, ciudad4) por interferencias en el nivel A0 porque Mover(ciudad2, ciudad3) tiene como precondición la proposición (Camion-en camion1 ciudad2) y la acción mover(ciudad2, ciudad4) produce un efecto que niega esta proposición.

◊ Necesidades competitivas

1º Mover(ciudad2, ciudad1)en nivel A2 es mutuamente exclusiva con Mover(ciudad1, ciudad2) por necesidades competitivas, porque sus precondiciones son mutex (Camion-en camion1 ciudad2) y su negación)

2º Mover(ciudad2, ciudad1) en nivel A2 es mutuamente exclusiva con Mover(ciudad3, ciudad2) por necesidades competitivas, porque sus precondiciones son mutex (Camion-en camion1 ciudad2) y su negación)

◊ **Soporte inconsistente (tipo 1)**

1º Camion-en camion1 ciudad1 es mutex con $\neg(\text{Camion-en camion1 ciudad1})$ en nivel P1 soporte inconsistente (tipo 1), porque uno es la negación de otro

2º paquete-en paquete1 ciudad1 es mutex con $\neg(\text{paquete-en paquete1 ciudad1})$ en nivel P2 por soporte inconsistente (tipo 1), porque uno es la negación de otro.

◊ **Soporte inconsistente (tipo 2)**

1º Camion-en camion1 ciudad1 en nivel P1 es mutex con Camion-en camion1 ciudad3 por soporte inconsistente (tipo 2), porque la acción de persistencia de mover(ciudad2, ciudad1) es mutex con la acción mover(ciudad2, ciudad3) (interferencia, mover(ciudad2, ciudad1) tiene como precondición la proposición (Camion-en camion1 ciudad2) y la acción mover(ciudad2, ciudad3) produce un efecto que niega la proposición son mutex)

2º Camion-en camion1 ciudad3 en nivel P1 es mutex con Camion-en camion1 ciudad4 por soporte inconsistente (tipo 2), porque la acción de persistencia de mover(ciudad2, ciudad3) es mutex con la acción mover(ciudad2, ciudad4) (interferencia, mover(ciudad2, ciudad3) tiene como precondición la proposición (Camion-en camion1 ciudad2) y la acción mover(ciudad2, ciudad4) produce un efecto que niega la proposición son mutex, ademas el resto de acciones que tienen como precondición o efecto lo evaluado en la interferencia tambien cumplen esa regla)

5. Conclusión

El desarrollo de esta actividad ha permitido aplicar técnicas fundamentales de planificación en inteligencia artificial. A través del modelado en PDDL se ha representado un problema logístico de transporte con restricciones reales, y mediante la búsqueda heurística A* se ha encontrado un plan óptimo que satisface el objetivo planteado.

La heurística utilizada ha resultado ser tanto admisible como consistente, lo que ha garantizado la eficacia del algoritmo A* durante la resolución del problema. Además, la construcción del grafo de planificación (GraphPlan) ha permitido identificar relaciones mutex entre acciones y proposiciones, lo que aporta una visión más profunda sobre la dinámica de exclusión en los estados posibles.

Esta actividad no solo refuerza el conocimiento sobre lenguajes de planificación y algoritmos heurísticos, sino que también entrena habilidades clave para abordar problemas complejos de toma de decisiones en entornos reales.