# PRÁCTICA LABORATORIO 1

Planificación automática

## OBJETIVOS:

Los objetivos de esta práctica son los siguientes:

- Poner en práctica el modelado de problemas de planificación utilizando un lenguaje formal que los planificadores automáticos entiendan (PDDL)
- Adquirir cierta idea de las capacidades y limitaciones de las tecnologías actuales de planificación independiente del dominio

Ignacio Peñalver Martín
Francisco González Velasco
Jaime Díez Buendía

Universidad de Alcalá

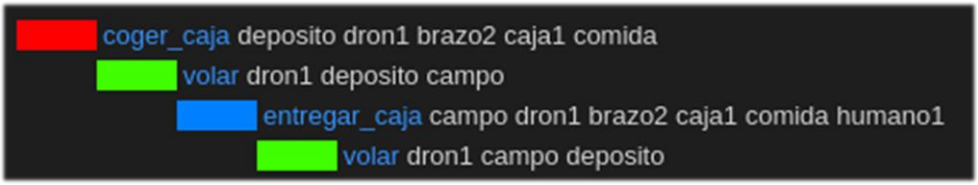
# ÍNDICE

# Parte 1 – Panificación Clásica con PDDL

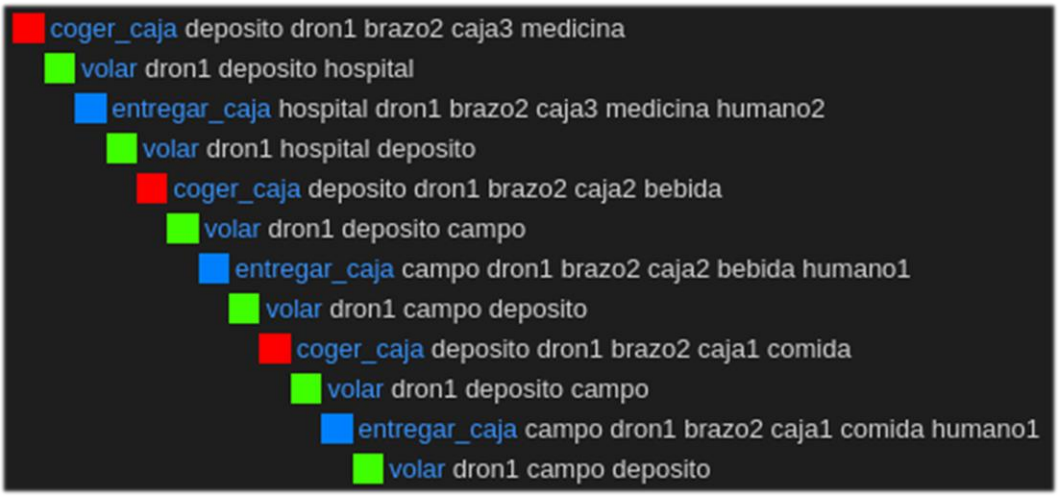
## Ejercicio 1.1: Logística de servicio de emergencias, versión inicial.

Para esta primera parte de la práctica, hemos creado un domino con tres acciones, las cuales son: volar, coger_caja, y entregar_caja. En el caso de volar, el dron puede volar con caja o sin caja. Y en cuanto a los brazos puede usar el que quiera para coger la caja, pero si la coje la cajaX con el brazo1, entregará la cajaX con el brazo1.

- Problema 1: Una persona y una caja




- Problema 2: Dos personas y tres cajas



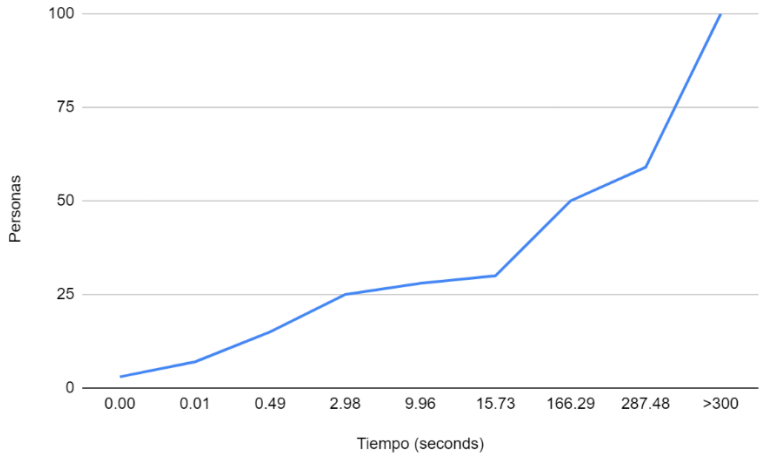

## Ejercicio 1.2: Generador de problemas en Python

**1.2. Genera problemas de complejidad creciente manteniendo -d 1 -r 0 y los parámetros -l, -p, -c y -g con un mismo valor (ej: ./generate-problem.py -d 1 -r 0 -l 5 -p 5 -c 5 -g 5) y ve probando a resolverlos con el planificador ff. ¿Hasta qué tamaño de problema es capaz de resolver disponiendo de un tiempo de aproximadamente 10 segundos? ¿Y si se dispone de 5 minutos máximo?**

A continuación, evaluaremos los tiempos (en segundos) con el planificador **fast forward** (ff) aumentando el número de personas en cada prueba, como vemos en la siguiente tabla.

| Personas | 3 | 7 | 15 | 25 | 28 | 30 | 50 | 59 | 100 |
|---|---|---|---|---|---|---|---|---|---|
| Tiempo (seconds) | 0.00 | 0.01 | 0.49 | 2.98 | 9.96 | 15.73 | 166.29 | 287.48 | >300 |

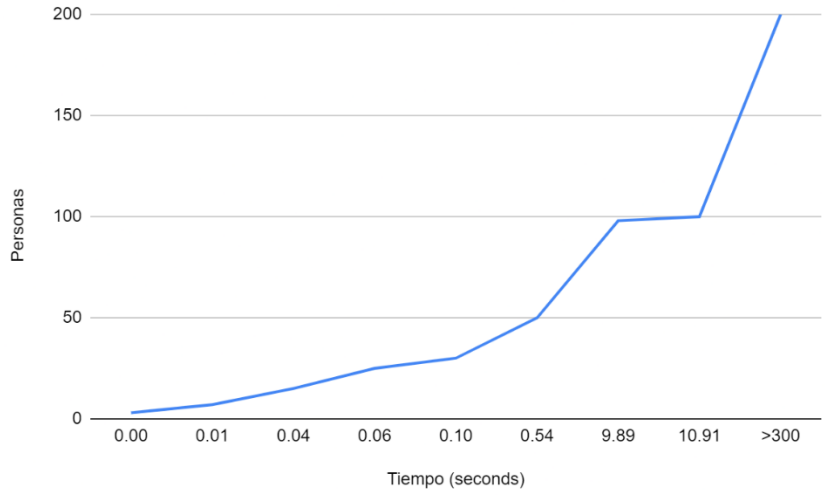
A partir de las pruebas realizadas y los datos obtenidos, como podemos ver en la tabla, para un tiempo menor de 10 segundos, el planificador puede resolver un problema con hasta 28 personas. Seguidamente, para un tiempo menor de 5 minutos, un problema con 59 personas es el máximo que ha podido lograr resolver (crece con un orden de x^3 aproximadamente).




El siguiente planificador al que vamos a evaluar su rendimiento es el **lpg-td** y al igual que antes iremos incrementando el número de personas cada vez.

| Personas | 3 | 7 | 15 | 25 | 30 | 50 | 98 | 100 | 200 |
|---|---|---|---|---|---|---|---|---|---|
| Tiempo (seconds) | 0.00 | 0.01 | 0.04 | 0.06 | 0.10 | 0.54 | 9.89 | 10.91 | >300 |

Podemos ver que este segundo planificador es sustancialmente mejor que el primero, ya que en un mismo tiempo (10s.) sube de poder resolver un problema de 59 personas a 98.



## Ejercicio 1.3: Comparativa rendimientos planificadores

A continuación, se muestra una tabla con el mayor tamaño que puede resolver cada uno de los planificadores en 10 segundos:

| | TAMAÑO | TIEMPO | NºACCIONES |
|---|---|---|---|
| LPG-TD | 90 | 9.7(24) | 370 |
| SGPLAN40 | 70 | 9.4 | 306 |
| SATPLAN | error | error | error |
| FASTDOWNWARD | 90 | 9.8 | 353 |

Se puede ver que el planificador que realiza el tamaño de problema más grande en 10 segundos es el FASTDOWNWARD junto con el LPG-TD.

PLANES:

- LPG-TD

  El plan realizado ha hecho 370 acciones para un tamaño de 90. En el plan se cogen varias cajas seguidas, pero a veces vuela innecesariamente.

- SGPLAN40

  El plan elaborado cuenta con 306 acciones para un tamaño de 70. Puede ser más optimo debido a que no entrega dos cajas seguidas.

- SATPLAN

*A pesar de intentarlo con todos los equipos disponibles e instalar el planificador una y otra vez, no hemos logrado que funcione:*

```
Memory limit exceeded: bb system call, OR Time limit exceeded: bb system call, OR bb internal error
---SatPlan Version: 1.1

bb: parsing domain file
domain 'PL1_DOMAIN' defined
 ... done.
bb: parsing problem file
problem 'DRONE_PROBLEM_D1_L80_P80_C80_G80_CT3' defined
 ... done.


NO MEMORY in file instantiateII.c:393


EXIT: Out of memory (Check_PTR)
```

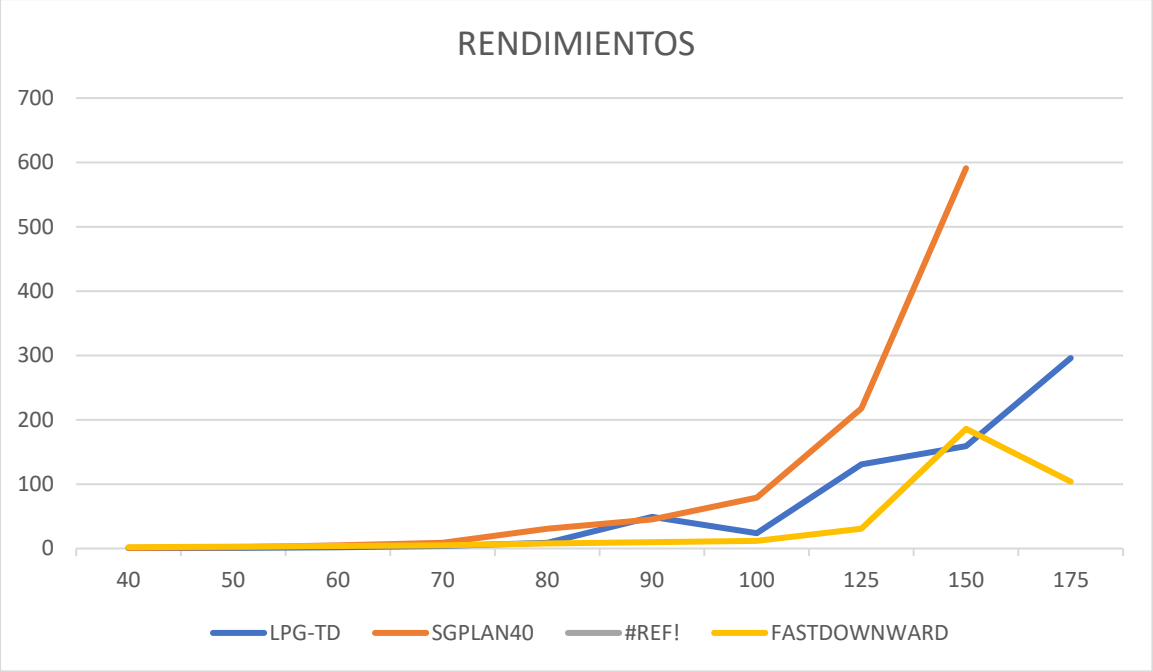
- FASTDOWNWARD

  El plan elaborado tiene 353 acciones para un tamaño de 90. El plan es óptimo porque entrega dos cajas seguidas al igual que las recoge.

Por otro lado, se muestra una tabla con el mayor tamaño que puede resolver cada uno de los planificadores en 300 segundos

| | TAMAÑO | TIEMPO | NºACCIONES | ÓPTIMO |
|---|---|---|---|---|
| LPG-TD | 150 | 296.3 | 568 | NO |
| SGPLAN40 | 125 | 273.1 | 556 | NO |
| SATPLAN | error | error | error | error |
| FASTDOWNWARD | 190 | 298.5 | 743 | SI |
| FF | 47 | 280.8 | 185 | SI |

A raíz de los resultados se ha realizado una gráfica para comparar los rendimientos de los diferentes planificadores:




Se puede ver que el planificador SGPLAN40 no es rápido ni óptimo en comparación con los demás. El planificador que más rápido y óptimo realiza los planes es el planificador FASTDOWNWARD, además de conseguir un plan de un problema más grande en el mismo tiempo que el resto de los planificadores.

Una cosa a destacar son los planes óptimos del planificador ff aunque tarda mucho en realizarlo y el tamaño del problema que se realiza es reducido.

# Parte 2 – Planificación con Números y Optimización

## Ejercicio 2.1: Servicio de emergencias, transportadores

**Apartado 3:**
En este apartado vamos a probar los distintos planificadores para el problema de la parte 2, que incluye los nums sin implementar fluents.

**Metricff:**
Si ejecutamos el problema pl1_2_problem.pddl tal cual obtenemos el siguiente plan:

```
$ ./metricff -o /home/…/pl1_2_domain.pddl -f /home/…/pl1_2_problem.pddl -s 1
```

```
ff: found legal plan as follows
step    0: COGER-CAJA DEPOSITO DRON1 CAJA6 BEBIDA
        1: VOLAR DRON1 DEPOSITO CAMPO
        2: ENTREGAR-CAJA-DRON HUMANO2 CAMPO DRON1 CAJA6 BEBIDA
        3: VOLAR DRON1 CAMPO DEPOSITO
        4: COGER-CAJA DEPOSITO DRON1 CAJA5 MEDICINAS
        5: VOLAR DRON1 DEPOSITO CAMPO
        6: ENTREGAR-CAJA-DRON HUMANO1 CAMPO DRON1 CAJA5 MEDICINAS
        7: VOLAR DRON1 CAMPO DEPOSITO
        8: COGER-CAJA DEPOSITO DRON1 CAJA4 COMIDA
        9: VOLAR DRON1 DEPOSITO CAMPO
       10: ENTREGAR-CAJA-DRON HUMANO1 CAMPO DRON1 CAJA4 COMIDA
       11: VOLAR DRON1 CAMPO DEPOSITO
       12: COGER-CAJA DEPOSITO DRON1 CAJA3 MEDICINAS
       13: VOLAR DRON1 DEPOSITO TEATRO
       14: ENTREGAR-CAJA-DRON HUMANO4 TEATRO DRON1 CAJA3 MEDICINAS
       15: VOLAR DRON1 TEATRO DEPOSITO
       16: COGER-CAJA DEPOSITO DRON1 CAJA2 BEBIDA
       17: VOLAR DRON1 DEPOSITO FIESTA
       18: ENTREGAR-CAJA-DRON HUMANO5 FIESTA DRON1 CAJA2 BEBIDA
       19: VOLAR DRON1 FIESTA DEPOSITO
       20: COGER-CAJA DEPOSITO DRON1 CAJA1 COMIDA
       21: VOLAR DRON1 DEPOSITO FIESTA
       22: ENTREGAR-CAJA-DRON HUMANO3 FIESTA DRON1 CAJA1 COMIDA
       23: VOLAR DRON1 FIESTA DEPOSITO
```

Si ejecutamos el problema pl1_2_problem.pddl forzando a que use el trasportador(es decir, eliminando la acción "volar") obtenemos el siguiente plan, (para demostrar que los nums están bien implementados):

```
$ ./metricff -o /home/…/pl1_2_domain.pddl -f /home/…/pl1_2_problem.pddl -s 1
```

```
ff: found legal plan as follows
step    0: COGER-CAJA DEPOSITO DRON1 CAJA6 BEBIDA
        1: PONER-CAJA-EN-TRANSPORTADOR DEPOSITO DRON1 CAJA6 TRANSPORTADOR1 N0 N1
        2: COGER-CAJA DEPOSITO DRON1 CAJA5 MEDICINAS
        3: PONER-CAJA-EN-TRANSPORTADOR DEPOSITO DRON1 CAJA5 TRANSPORTADOR1 N1 N2
        4: COGER-CAJA DEPOSITO DRON1 CAJA4 COMIDA
        5: PONER-CAJA-EN-TRANSPORTADOR DEPOSITO DRON1 CAJA4 TRANSPORTADOR1 N2 N3
        6: COGER-TRANSPORTADOR DEPOSITO DRON1 TRANSPORTADOR1
        7: MOVER-TRANSPORTADOR DEPOSITO CAMPO TRANSPORTADOR1 DRON1
        8: SOLTAR-TRANSPORTADOR CAMPO DRON1 TRANSPORTADOR1
        9: COGER-CAJA-DEL-TRANSPORTADOR CAMPO DRON1 CAJA4 MEDICINAS TRANSPORTADOR1 N2 N3
       10: ENTREGAR-CAJA-DRON HUMANO1 CAMPO DRON1 CAJA4 COMIDA
       11: COGER-CAJA-DEL-TRANSPORTADOR CAMPO DRON1 CAJA5 MEDICINAS TRANSPORTADOR1 N1 N2
       12: ENTREGAR-CAJA-DRON HUMANO1 CAMPO DRON1 CAJA5 MEDICINAS
       13: COGER-CAJA-DEL-TRANSPORTADOR CAMPO DRON1 CAJA6 MEDICINAS TRANSPORTADOR1 N0 N1
       14: ENTREGAR-CAJA-DRON HUMANO2 CAMPO DRON1 CAJA6 BEBIDA
       15: COGER-TRANSPORTADOR CAMPO DRON1 TRANSPORTADOR1
       16: MOVER-TRANSPORTADOR CAMPO DEPOSITO TRANSPORTADOR1 DRON1
       17: SOLTAR-TRANSPORTADOR DEPOSITO DRON1 TRANSPORTADOR1
       18: COGER-CAJA DEPOSITO DRON1 CAJA3 MEDICINAS
       19: PONER-CAJA-EN-TRANSPORTADOR DEPOSITO DRON1 CAJA3 TRANSPORTADOR1 N0 N1
       20: COGER-CAJA DEPOSITO DRON1 CAJA2 BEBIDA
       21: PONER-CAJA-EN-TRANSPORTADOR DEPOSITO DRON1 CAJA2 TRANSPORTADOR1 N1 N2
       22: COGER-CAJA DEPOSITO DRON1 CAJA1 COMIDA
       23: PONER-CAJA-EN-TRANSPORTADOR DEPOSITO DRON1 CAJA1 TRANSPORTADOR1 N2 N3
       24: COGER-TRANSPORTADOR DEPOSITO DRON1 TRANSPORTADOR1
       25: MOVER-TRANSPORTADOR DEPOSITO FIESTA TRANSPORTADOR1 DRON1
       26: SOLTAR-TRANSPORTADOR FIESTA DRON1 TRANSPORTADOR1
       27: COGER-CAJA-DEL-TRANSPORTADOR FIESTA DRON1 CAJA1 MEDICINAS TRANSPORTADOR1 N2 N3
       28: ENTREGAR-CAJA-DRON HUMANO3 FIESTA DRON1 CAJA1 COMIDA
       29: COGER-CAJA-DEL-TRANSPORTADOR FIESTA DRON1 CAJA2 MEDICINAS TRANSPORTADOR1 N1 N2
       30: ENTREGAR-CAJA-DRON HUMANO5 FIESTA DRON1 CAJA2 BEBIDA
       31: COGER-TRANSPORTADOR FIESTA DRON1 TRANSPORTADOR1
       32: MOVER-TRANSPORTADOR FIESTA DEPOSITO TRANSPORTADOR1 DRON1
       33: MOVER-TRANSPORTADOR DEPOSITO TEATRO TRANSPORTADOR1 DRON1
       34: SOLTAR-TRANSPORTADOR TEATRO DRON1 TRANSPORTADOR1
       35: COGER-CAJA-DEL-TRANSPORTADOR TEATRO DRON1 CAJA3 MEDICINAS TRANSPORTADOR1 N0 N1
       36: ENTREGAR-CAJA-DRON HUMANO4 TEATRO DRON1 CAJA3 MEDICINAS
       37: COGER-TRANSPORTADOR TEATRO DRON1 TRANSPORTADOR1
       38: MOVER-TRANSPORTADOR TEATRO DEPOSITO TRANSPORTADOR1 DRON1
```

**FastForward:** En este caso observamos que no utiliza el transportador, pero realiza correctamente las entregas.

```
ff: found legal plan as follows

step    0: COGER-CAJA DEPOSITO DRON1 CAJA3 MEDICINAS
        1: VOLAR DRON1 DEPOSITO TEATRO
        2: ENTREGAR-CAJA-DRON HUMANO4 TEATRO DRON1 CAJA3 MEDICINAS
        3: VOLAR DRON1 TEATRO DEPOSITO
        4: COGER-CAJA DEPOSITO DRON1 CAJA5 MEDICINAS
        5: VOLAR DRON1 DEPOSITO CAMPO
        6: ENTREGAR-CAJA-DRON HUMANO1 CAMPO DRON1 CAJA5 MEDICINAS
        7: VOLAR DRON1 CAMPO DEPOSITO
        8: COGER-CAJA DEPOSITO DRON1 CAJA1 COMIDA
        9: VOLAR DRON1 DEPOSITO FIESTA
       10: ENTREGAR-CAJA-DRON HUMANO3 FIESTA DRON1 CAJA1 COMIDA
       11: VOLAR DRON1 FIESTA DEPOSITO
       12: COGER-CAJA DEPOSITO DRON1 CAJA4 COMIDA
       13: VOLAR DRON1 DEPOSITO CAMPO
       14: ENTREGAR-CAJA-DRON HUMANO1 CAMPO DRON1 CAJA4 COMIDA
       15: VOLAR DRON1 CAMPO DEPOSITO
       16: COGER-CAJA DEPOSITO DRON1 CAJA2 BEBIDA
       17: VOLAR DRON1 DEPOSITO FIESTA
       18: ENTREGAR-CAJA-DRON HUMANO5 FIESTA DRON1 CAJA2 BEBIDA
       19: VOLAR DRON1 FIESTA DEPOSITO
       20: COGER-CAJA DEPOSITO DRON1 CAJA6 BEBIDA
       21: VOLAR DRON1 DEPOSITO CAMPO
       22: ENTREGAR-CAJA-DRON HUMANO2 CAMPO DRON1 CAJA6 BEBIDA
       23: VOLAR DRON1 CAMPO DEPOSITO
```

**LPG-TD:** Este planificador tampoco utiliza el transportador.

```
Plan computed:
   Time: (ACTION) [action Duration; action Cost]
 0.0000: (COGER-CAJA DEPOSITO DRON1 CAJA2 BEBIDA) [D:1.00; C:1.00]
 1.0000: (VOLAR DRON1 DEPOSITO TEATRO) [D:1.00; C:1.00]
 2.0000: (VOLAR DRON1 TEATRO CAMPO) [D:1.00; C:1.00]
 3.0000: (ENTREGAR-CAJA-DRON HUMANO2 CAMPO DRON1 CAJA2 BEBIDA) [D:1.00; C:1.00]
 4.0000: (VOLAR DRON1 CAMPO FIESTA) [D:1.00; C:1.00]
 5.0000: (VOLAR DRON1 FIESTA DEPOSITO) [D:1.00; C:1.00]
 6.0000: (COGER-CAJA DEPOSITO DRON1 CAJA1 COMIDA) [D:1.00; C:1.00]
 7.0000: (VOLAR DRON1 DEPOSITO CAMPO) [D:1.00; C:1.00]
 8.0000: (ENTREGAR-CAJA-DRON HUMANO1 CAMPO DRON1 CAJA1 COMIDA) [D:1.00; C:1.00]
 9.0000: (VOLAR DRON1 CAMPO DEPOSITO) [D:1.00; C:1.00]
 10.0000: (COGER-CAJA DEPOSITO DRON1 CAJA5 MEDICINAS) [D:1.00; C:1.00]
 11.0000: (VOLAR DRON1 DEPOSITO CAMPO) [D:1.00; C:1.00]
 12.0000: (ENTREGAR-CAJA-DRON HUMANO1 CAMPO DRON1 CAJA5 MEDICINAS) [D:1.00; C:1.00]
 13.0000: (VOLAR DRON1 CAMPO DEPOSITO) [D:1.00; C:1.00]
 14.0000: (COGER-CAJA DEPOSITO DRON1 CAJA6 BEBIDA) [D:1.00; C:1.00]
 15.0000: (VOLAR DRON1 DEPOSITO FIESTA) [D:1.00; C:1.00]
 16.0000: (ENTREGAR-CAJA-DRON HUMANO5 FIESTA DRON1 CAJA6 BEBIDA) [D:1.00; C:1.00]
 17.0000: (VOLAR DRON1 FIESTA DEPOSITO) [D:1.00; C:1.00]
 18.0000: (COGER-CAJA DEPOSITO DRON1 CAJA4 COMIDA) [D:1.00; C:1.00]
 19.0000: (VOLAR DRON1 DEPOSITO FIESTA) [D:1.00; C:1.00]
 20.0000: (ENTREGAR-CAJA-DRON HUMANO3 FIESTA DRON1 CAJA4 COMIDA) [D:1.00; C:1.00]
 21.0000: (VOLAR DRON1 FIESTA DEPOSITO) [D:1.00; C:1.00]
 22.0000: (COGER-CAJA DEPOSITO DRON1 CAJA3 MEDICINAS) [D:1.00; C:1.00]
 23.0000: (VOLAR DRON1 DEPOSITO TEATRO) [D:1.00; C:1.00]
 24.0000: (ENTREGAR-CAJA-DRON HUMANO4 TEATRO DRON1 CAJA3 MEDICINAS) [D:1.00; C:1.00]
 25.0000: (VOLAR DRON1 TEATRO DEPOSITO) [D:1.00; C:1.00]
```

**Satplan:** Este planificador no nos da ningún plan como solución al problema, nos produce a los 3 integrantes del grupo un error de ***system call***.

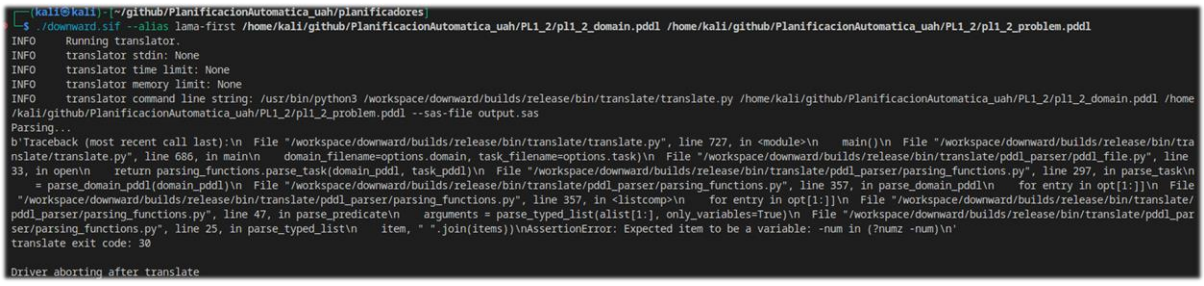
```
Memory limit exceeded: bb system call, OR Time limit exceeded: bb system call, OR bb internal error
---SatPlan Version: 1.1

bb: parsing domain file
domain 'PL1_DOMAIN' defined
 ... done.
bb: parsing problem file
problem 'DRONE_PROBLEM_D1_L80_P80_C80_G80_CT3' defined
 ... done.


NO MEMORY in file instantiateII.c:393


EXIT: Out of memory (Check_PTR)
```

**Sgplan40:** Este planificador tampoco coge el transportador en su plan.

```
0.010:  (COGER-CAJA DEPOSITO DRON1 CAJA4 COMIDA)[0.000]
0.020:  (VOLAR DRON1 DEPOSITO CAMPO)[0.000]
0.030:  (ENTREGAR-CAJA-DRON HUMANO1 CAMPO DRON1 CAJA4 COMIDA)[0.000]
0.040:  (VOLAR DRON1 CAMPO DEPOSITO)[0.000]
0.050:  (COGER-CAJA DEPOSITO DRON1 CAJA1 COMIDA)[0.000]
0.060:  (VOLAR DRON1 DEPOSITO CAMPO)[0.000]
0.070:  (VOLAR DRON1 CAMPO FIESTA)[0.000]
0.080:  (ENTREGAR-CAJA-DRON HUMANO3 FIESTA DRON1 CAJA1 COMIDA)[0.000]
0.090:  (VOLAR DRON1 FIESTA DEPOSITO)[0.000]
0.100:  (COGER-CAJA DEPOSITO DRON1 CAJA5 MEDICINAS)[0.000]
0.110:  (VOLAR DRON1 DEPOSITO TEATRO)[0.000]
0.120:  (ENTREGAR-CAJA-DRON HUMANO4 TEATRO DRON1 CAJA5 MEDICINAS)[0.000]
0.130:  (VOLAR DRON1 TEATRO CAMPO)[0.000]
0.140:  (VOLAR DRON1 CAMPO DEPOSITO)[0.000]
0.150:  (COGER-CAJA DEPOSITO DRON1 CAJA3 MEDICINAS)[0.000]
0.160:  (VOLAR DRON1 DEPOSITO CAMPO)[0.000]
0.170:  (ENTREGAR-CAJA-DRON HUMANO1 CAMPO DRON1 CAJA3 MEDICINAS)[0.000]
0.180:  (VOLAR DRON1 CAMPO DEPOSITO)[0.000]
0.190:  (COGER-CAJA DEPOSITO DRON1 CAJA6 BEBIDA)[0.000]
0.200:  (VOLAR DRON1 DEPOSITO CAMPO)[0.000]
0.210:  (ENTREGAR-CAJA-DRON HUMANO2 CAMPO DRON1 CAJA6 BEBIDA)[0.000]
0.220:  (VOLAR DRON1 CAMPO FIESTA)[0.000]
0.230:  (VOLAR DRON1 FIESTA DEPOSITO)[0.000]
0.240:  (COGER-CAJA DEPOSITO DRON1 CAJA2 BEBIDA)[0.000]
0.250:  (VOLAR DRON1 DEPOSITO FIESTA)[0.000]
0.260:  (ENTREGAR-CAJA-DRON HUMANO5 FIESTA DRON1 CAJA2 BEBIDA)[0.000]
0.270:  (VOLAR DRON1 FIESTA DEPOSITO)[0.000]
```

**FastDownward:** Este planificador devuelve un error a la hora de compilación (fatal-translate-error 30), el cual significa que el método translate.py del downward no se ejecuta correctamente.

*ErrorFastDownward: Nos hemos instalado y compilado de cero el planificador downward, tanto de tu repositorio como de la página oficial de este planificador, y aun así nos ha seguido generando el mismo error a los tres. Por lo tanto, no hemos podido comprobar el plan que crea dicho planificador con el dominio y los problemas de la parte2.*



## Ejercicio 2.2: Servicio de emergencias, costes de acción

**Apartado 8:**

**a)** El planificador Metric-FF utiliza el algoritmo de búsqueda **A***, este es un algoritmo de búsqueda de grafos que combina la búsqueda primero en profundidad con una heurística que estima el costo restante para alcanzar la meta.

El parámetro **-w** es el peso de la heurística, que ajusta la importancia relativa de la heurística frente al costo real.

Si se establece el valor de **-w** en 1, el planificador realiza una búsqueda de costo uniforme en lugar de utilizar la heurística. En este caso, el planificador considera todos los nodos del grafo y no toma en cuenta la información heurística que puede acelerar la búsqueda.

En el caso de dar un valor muy alto a **-w,** el planificador pasará por alto las soluciones que tienen un costo más bajo en términos reales. Esto puede provocar que el planificador se atasque y no encuentre la solución óptima en términos de la función de costo especificada.

**b)** LAMA es un tipo de planificador basado en heurísticas, específicamente un planificador que utiliza búsqueda A* y una heurística admisible y consistente para guiar la búsqueda.

Cuando llamamos a downward.sif con la opción --alias lama-first, el planificador utilizará LAMA **como primera opción**. LAMA primero utiliza una técnica de expansión de estados basada en heurísticas y luego, si falla, recurre a una búsqueda basada en SAT.

Por otro lado, si llamamos a downward.sif con la opción --alias lama, el planificador utilizará LAMA **como única opción**. En este caso, LAMA se ejecutará con su técnica de expansión de estados basada en heurísticas sin recurrir a una búsqueda basada en SAT.

**c)** BJOLP es un tipo de planificador basado en técnicas de optimización lineal entera mixta (MILP). En lugar de realizar una búsqueda explícita en el espacio de estados, como lo hacen los planificadores basados en heurísticas, BJOLP formula el problema de planificación como un problema de optimización lineal entera y lo resuelve utilizando técnicas de optimización.

En general, se espera que BJOLP sea más eficiente que los planificadores basados en heurísticas en problemas con un gran número de variables y restricciones.

### ¿Qué tamaño de problema es capaz de resolver en menos de 5 minutos?

Debido al *ErrorFastDownward* no hemos podido comprobar lo que puede resolver este planificador.

Los planificadores basados en heurísticas, como LAMA, suelen ser más eficientes que los planificadores basados en optimización, como BJOLP, para problemas de planificación con un horizonte temporal corto y una cantidad limitada de acciones como es el caso de nuestro problema.

**d)** FDSS es un tipo de planificador basado en búsqueda. En particular, utiliza una estrategia de búsqueda basada en la exploración del espacio de estados mediante la técnica de búsqueda en árbol (tree search), combinando técnicas de búsqueda heurísticas y no heurísticas. El algoritmo de búsqueda utilizado por FDSS se conoce como búsqueda de saltos de forward (Forward Jumping Search).

Por otro lado, un planificador de tipo "portfolio" es un planificador que combina varios planificadores diferentes en una única herramienta. Cada uno de los planificadores en el portafolio puede tener diferentes enfoques, estrategias de búsqueda y heurísticas. En tiempo de ejecución, el planificador selecciona uno o varios de los planificadores del portafolio y los utiliza para resolver el problema de planificación. La idea detrás de un planificador de tipo "portfolio" es que, al combinar diferentes planificadores, se puede aprovechar la fortaleza de cada uno de ellos y mejorar el rendimiento en general.

### ¿Cuánto tarda en resolver el problema generado en el apartado c, en el que BJOLP tardaba 5 minutos aproximadamente?

Debido al *ErrorFastDownward* no hemos podido comprobar lo que puede resolver este planificador.

# Parte 3 – Planificación con Concurrencia

## Ejercicio 3.1: Acciones concurrentes en gestión de emergencias

**Pensar qué acciones no deberían poder ejecutar dos drones en paralelo:**

Como bien nos indica el enunciado, dos drones no podrían coger la misma caja a la vez, lo que se extiende al transportador, con el que pasa lo mismo.

Por lo tanto, suponiendo que sólo hubiera un transportador para los dos drones, uno tendría que dedicarse a entregar las cajas en las localizaciones más solicitadas haciendo uso del mismo y, el otro llevaría las cajas de una en una a las ubicaciones en las que hay menos personas necesitadas. Cabe destacar que digamos el dron1 no haría siempre una cosa y el dron2 otra. Estando los dos drones en el depósito, el que primero cogiese el transportador haría uso de este y el otro tendría que conformarse con una caja solo.

Podríamos pensar que ambos drones pueden ayudarse e ir llenando juntos el transportador, pero, esto no sería posible ya que modificamos el llenado de este dentro de la función y sino, entrarían en juego las condiciones de carrera. Imaginemos que el transportador está en 3 cajas de capacidad y 4 es el máximo. Si ambos drones ejecutasen la función a la vez, ambos verían que cabe una caja y ambos meterían una caja y actualizarían luego el número de cajas a 4, siendo esto falso ya que realmente habría 5. Exactamente lo mismo pasaría para coger cajas del transportador. Al no utilizar cerrojos u otras estructuras que aseguren la integridad referencial de los datos, no nos permite que esto suceda.

Al igual que hemos mencionado antes que tan sólo un dron podría coger el transportador en un momento dado, esto se extiende a la acción mover-transportador.

Obviamente sólo un dron puede soltar el transportador a la vez ya que tan sólo uno puede tenerlo cogido en un cierto momento de tiempo.

El resto de las acciones como coger una caja (siempre que sea distinta) y entregar una caja, sí que podrían ser realizadas a la vez. Asimismo, sí se podían entregar dos cajas a la misma persona a la vez, ya que cada una de ellas mitigaría una necesidad del humano distinta y podrían pasar las dos a estar satisfechas.

## Ejercicio 3.2: Implementación y pruebas de concurrencia

El único plan que genera el planificador optic en menos de un minuto, con dos drones y dos transportadores es:

```
; Plan found with metric 100.000
; Theoretical reachable cost 100.000
; States evaluated so far: 271
; States pruned based on pre-heuristic cost lower bound: 0
; Time 0.62
0.000: (coger-caja deposito dron1 caja3 medicinas)  [0.001]
0.000: (coger-caja deposito dron2 caja5 medicinas)  [0.001]
0.001: (volar dron1 deposito teatro)  [15.000]
0.001: (volar dron2 deposito campo)  [5.000]
5.002: (entregar-caja-dron humano1 campo dron2 caja5 medicinas)  [0.001]
5.003: (volar dron2 campo deposito)  [5.000]
10.004: (coger-caja deposito dron2 caja1 comida)  [0.001]
10.005: (volar dron2 deposito campo)  [5.000]
15.002: (entregar-caja-dron humano4 teatro dron1 caja3 medicinas)  [0.001]
15.003: (volar dron1 teatro deposito)  [15.000]
15.006: (entregar-caja-dron humano1 campo dron2 caja1 comida)  [0.001]
15.007: (volar dron2 campo deposito)  [5.000]
20.008: (coger-caja deposito dron2 caja2 bebida)  [0.001]
20.009: (volar dron2 deposito campo)  [5.000]
25.010: (entregar-caja-dron humano2 campo dron2 caja2 bebida)  [0.001]
25.011: (volar dron2 campo deposito)  [5.000]
30.004: (coger-caja deposito dron1 caja4 comida)  [0.001]
30.005: (volar dron1 deposito fiesta)  [10.000]
30.012: (coger-caja deposito dron2 caja6 bebida)  [0.001]
30.013: (volar dron2 deposito fiesta)  [10.000]
40.006: (entregar-caja-dron humano3 fiesta dron1 caja4 comida)  [0.001]
40.007: (volar dron1 fiesta deposito)  [10.000]
40.014: (entregar-caja-dron humano5 fiesta dron2 caja6 bebida)  [0.001]
40.015: (volar dron2 fiesta deposito)  [10.000]
```

Como se puede observar hace un uso correcto de ambos drones, y el reparto de cajas ya que cada dron solo entrega la caja que coje.

El próximo plan lo genera a los 2 minutos 21 segundos.