PRÁCTICA 1

PARTE 1

Marcos Fúster Peña

Daniel de la Fuente Sánchez

Pablo Tirado Barragán

M emoria de la Práctica 1 – Planificación Clásica con PDDL

1. Introducción

1.1 Objetivo de la Práctica

Esta práctica tiene como objetivo aplicar distintos modelados de problemas de planificación automática utilizando PDDL. Se busca entender las capacidades, funcionalidades y limitaciones de los distintos tipos de modelos de planificación y su aplicabilidad en escenarios reales.

Este proyecto centrará los problemas en la gestión de un sistema de atención de emergencia de transporte de mercancía a través de drones. Posteriormente se probará el modelo con distintos planificadores para evaluar su rendimiento.

Los objetivos específicos de la práctica incluyen:

- Modelar el problema de planificación en PDDL clásico con restricciones STRIPS[1].
- Crear un generador de problemas aleatorio donde se le pasará la cantidad de elementos que participarán en la creación.
- Evaluar el rendimiento de distintos planificadores automáticos.

1.2 Estructura de la Documentación

Esta memoria está estructurada en las siguientes secciones:

- **Introducción**: Presenta los objetivos de la práctica y una visión general de la documentación.
- **Desarrollo de la Práctica**: Explica el modelado del dominio en PDDL, la generación de problemas en Python y la evaluación de los planificadores.
- **Resultados y Análisis**: Muestra los experimentos realizados y una comparación del rendimiento de distintos planificadores.
- **Conclusiones**: Resume los hallazgos clave y propone mejoras futuras.

[1] STRIPS: Restricciones en problemas de planificación. Esto se aplica como: Tiene precondiciones, que de ben cumplirse antes de ejecutar una acción; efectos, que son los cambios que aplican las funciones; sin condiciones negativas, no se puede definir una condición como opuesta a una existente; sin efectos condicionales, no se puede definir bifurcaciones en la toma de decisiones de una acción concreta; y no tiene cuantificadores ni expresiones complejas, expresiones como ∀ (para todo) o ∃ (existe).

1.3 Contenido del Proyecto

El proyecto incluye los siguientes archivos y directorios:

PDDL

- parte1/pddl/dominio-drones.pddl – Modelo del dominio de planificación.
- parte1/pddl/problema.pddl – Instancia de problema generado como prueba

Generador de Problemas

- parte1/src/generate_problem.py – Script en Python para la generación de problemas.
- parte1/src/test_planiffier.py – Script en Python para generar gráficos de eficiencia de los planificadores dados.

Planificadores

- planificadores/ff – Planificador Fast Forward (FF).
- planificadores/lpg-td – Planificador LPG-TD basado en búsqueda local.
- planificadores/sgplan40 – Planificador SGPLAN40 basado en descomposición de problemas.

ProblemasGenerados

- Almacenamiento de problemas generados por el generador de problemas

Resultados

- Almacenamiento de los resultados obtenidos en los ejecutables de generación y testeo

---

---

2. Desarrollo de la Práctica

2.1 Modelado del Problema en PDDL

- **Descripción del dominio**:

El dominio de esta primera parte es uno muy sencillo.

En primer lugar, hemos definido una serie de tipos para identificar a los distintos tipos de actores que formarán parte del problema. Estos tipos son:

- dron: Es el instrumento de transporte de mercancías y el protagonista del desarrollo de este problema de planificación.
- brazo: Componente del dron con el que este puede interactuar para llevar las cosas de un lado a otro. Es independiente al dron ya que de esta forma, incluso aplicando funcionalidades equivalentes, es capaz de ser una aplicación mucho más escalable.
- persona: Usuario solicitante de suministros. Se localizan en algún sitio y necesitan un tipo de suministro capaz de ser transportado por los drones.
- localización: Posición o lugar física por la que comprendemos la capacidad de un dron de relacionarse con el ambiente.
- caja: Instrumento de transporte de suministros.
- contenido: Suministros

Tras especificar los tipos, hemos creado unos predicados para permitir al sistema interactuar entre sí. Estos son los predicados:

- **dron-en** ?d – dron ?l – localización: Indica si un dron está en una posición
- **caja-en** ?c – caja
- **persona-en** ?p - persona ?l - localización: Indica si una persona está en una posición
- **sostiene** ?d - dron ?b - brazo ?c - caja: Indica si un dron sostiene una caja con un brazo.
- **brazo-libre** ?d - dron ?b - brazo: Indica si un brazo de un dron está libre.
- **necesita** ?p - persona ?t - contenido: Indica si una persona necesita un cierto contenido.
- **tiene** ?p - persona ?t - contenido: Indica si una persona posee un cierto contenido.
- **contiene** ?c - caja ?t - contenido: Indica si una caja contiene un cierto contenido.

Finalmente, para poder realizar los cambios en el sistema, debe haber acciones que causen dichos cambios. Para ello, hacemos uso de las acciones, que son las siguientes:

Acción: coger

Descripción

Un dron usa uno de sus brazos para recoger una caja en la misma localización.

Precondiciones

- El dron (?d) debe estar en la misma localización (?l) que la caja (?c).

- El brazo (?b) del dron debe estar libre.

Efectos

- El dron sostiene la caja con el brazo (sostiene ?d ?b ?c).
- La caja ya no está en la localización (not (caja-en ?c ?l)).
- El brazo utilizado ya no está libre (not (brazo-libre ?d ?b)).

Acción: volar

Descripción

El dron se desplaza de una localización a otra.

Precondiciones

- El dron (?d) debe estar en la localización de origen (?from).

Efectos

- El dron deja de estar en la localización de origen (not (dron-en ?d ?from)).
- El dron pasa a estar en la nueva localización (dron-en ?d ?to).

Acción: entregar

Descripción

Un dron entrega una caja a una persona que la necesita.

Precondiciones

- El dron (?d) debe sostener la caja (?c) con el brazo (?b).
- El dron (?d) y la persona (?p) deben estar en la misma localización (?l).
- La caja (?c) debe contener el contenido (?t).
- La persona (?p) debe necesitar el contenido (necesita ?p ?t).

Efectos

- La persona (?p) obtiene el contenido (tiene ?p ?t).
- El dron deja de sostener la caja (not (sostiene ?d ?b ?c)).
- La persona ya no necesita el contenido (not (necesita ?p ?t)).
- El brazo del dron queda libre (brazo-libre ?d ?b).

2.2 Generador de Problemas en Python

- **Descripción del código**:

Para crear problemas, hemos usado el esqueleto del generador proporcionado y hemos gestionado algunos cambios para adaptarlo al dominio que hemos creados.

Este generador de problemas aleatorio tiene en cuanta una serie de parámetros dados por el usuario de la aplicación y genera un problema con la cantidad de actores y acciones solicitadas de forma completamente aleatoria.

Para generar correctamente el problema, hay una serie de funciones auxiliares que apoyan a su creación.

LAS FUNCIONES AUXILIARES:

distance(location_coords, location_num1, location_num2): Función proporcionada por el esqueleto. Obtiene la distancia euclidiana entre dos ubicaciones. No la usaremos en esta parte del trabajo.

flight_cost(location_coords, location_num1, location_num2): Función proporcionada por el esqueleto. Caclula el coste de vuelo entre dos ubicaciones basado en la distancia. Llama a distance() para obtener la distancia entre ellos. Devuelve el valor de la acción basado en la distancia redondeada hacia arriba. No la usaremos en esta parte del trabajo.

setup_content_types(options): Función proporcionado por el esqueleto. Esta función auxiliar trata de generar de forma aleatoria el contendido de las cajas, siendo estos pasados como parámetro. Además, esta función se asegura de que al menos una caja tenga un elemento de los existentes en el sistema (en caso de haber comida y medicina, habrá una de cada en alguna de las cajas)

setup_location_coords(options): Función proporcionada por el esqueleto. Esta función asigna coordenadas aleatorias a las localizaciones (tipo hablado anteriormente). No la usaremos en esta parte del trabajo.

setup_person_needs(options, crates_with_contents): Genera necesidades aleatorias de contenido para las personas generadas en el problema. Esta función asegura de que no se le asignen más necesidades de las que debe cubrirse.

FUNCIÓN PRINCIPAL/MAIN:

La función principal permite, mediante la introducción de una serie de parámetros que indican los actores que formarán parte de un sistema, que este genere un problema aleatorio.

El programa es resistente a la no introducción de los parámetros, generando errores de ejecución indicando los parámetros necesarios para su correcta ejecución.

- **Ejecución del generador**:
    - Tipos de problemas generados y análisis de su dificultad.

Para ejecutar correctamente el generador, es necesario llamarlo por la termina. Esto se hace mediante la llamada del archivo de Python y es necesario pasarle obligatoriamente los siguiente parámetros:

- --drones: Seguido de un número, indica el número de drones que creará el problema
- --carriers: Seguido de un número, indica el número de brazos que tendrá el dron
- --locations: Seguido de un número, indica la cantidad de localizaciones que tendrá el problema
- --personas: Seguido de un número, indica la cantidad de localizaciones que tendrá el problema.
- --crates: Seguido de un número, indica la cantidad de las cajas que tendrá el problema
- --goals: Seguido de un número, indica la cantidad de problemas que generará el problema
- --output: Seguido de una ruta concreta, indica donde se guardará el archivo generado con el problema.

Y tras cada uno de esos parámetros, es necesario añadirle un número, que indicará la cantidad de ese tipo se crearán para el programa.

El problema se guarda en la carpeta problemasGenerados de la parte que le corresponda.

EJEMPLO DE USO:

python parte1\generadores\generadorAleatorio.py --drones 3 --carriers 2 --locations 5 --persons 4 --crates 6 --goals 4

Mediante el anterior comando, hemos llamado al programa desde la carpeta general del proyecto y genera un archivo tal que:

```
(define (problem drone_problem_d3_r2_l5_p4_c6_g4_ct2)
(:domain dominio-drones)
(:objects
    dron1 dron2 dron3 - dron
    deposito loc1 loc2 loc3 loc4 loc5 - localizacion
    caja1 caja2 caja3 caja4 caja5 caja6 - caja
    comida medicina - contenido
    pers1 pers2 pers3 pers4 - persona
    brazo1 brazo2 - brazo
)
(:init
    (dron-en dron1 deposito)
    (brazo-libre dron1 brazo1)
    (brazo-libre dron1 brazo2)
    (dron-en dron2 deposito)
    (brazo-libre dron2 brazo1)
    (brazo-libre dron2 brazo2)
    (dron-en dron3 deposito)
    (brazo-libre dron3 brazo1)
    (brazo-libre dron3 brazo2)
    (caja-en caja1 loc5) (contiene caja1 medicina)
    (caja-en caja2 loc4) (contiene caja2 medicina)
    (caja-en caja3 loc2) (contiene caja3 comida)
    (caja-en caja4 loc2) (contiene caja4 comida)
    (caja-en caja5 loc3) (contiene caja5 medicina)
    (caja-en caja6 loc2) (contiene caja6 medicina)
    (persona-en pers1 loc3)
    (necesita pers1 comida)
    (persona-en pers2 loc1)
    (persona-en pers3 loc4)
    (necesita pers3 comida)
    (persona-en pers4 loc1)
)
(:goal (and
    (dron-en dron1 deposito)
    (dron-en dron2 deposito)
    (dron-en dron3 deposito)
    (tiene pers1 comida)
    (tiene pers3 comida)
    ))
)
```

2.3 Ejemplo de problema:

El problema que vamos a tratar es el siguiente:

```
(define (problem problema)
  (:domain dominio-drones)

  (:objects
    d1 - dron
    b1 b2 - brazo
    p1 p2 p3 p4 - persona
    deposito l1 l2 l3 l4 - localizacion
    c1 c2 c3 c4 c5 c6 c7 - caja
    comida medicina agua - contenido
  )

  (:init
    ;; Ubicación inicial del dron y las cajas en el depósito
    (dron-en d1 deposito)
    (brazo-libre d1 b1)
    (brazo-libre d1 b2)

    (caja-en c1 deposito) (contiene c1 comida)
    (caja-en c2 deposito) (contiene c2 medicina)
    (caja-en c3 deposito) (contiene c3 comida)
    (caja-en c4 deposito) (contiene c4 comida)
    (caja-en c5 deposito) (contiene c5 medicina)
    (caja-en c6 deposito) (contiene c6 medicina)
    (caja-en c7 deposito) (contiene c7 comida)

    ;; Ubicación de las personas y sus necesidades
    (persona-en p1 l1) (necesita p1 comida)
    (persona-en p2 l2) (necesita p2 medicina)
    (persona-en p3 l3) (necesita p3 comida)
    (persona-en p4 l4) (necesita p4 comida))

  (:goal
    (and
      (tiene p1 comida)
      (tiene p2 medicina)
      (tiene p3 comida)
      (tiene p4 comida)))
)
```

Para entender el problema, vamos a desglosarlo y comentarlo:

En dicho problema, podemos identificar estos objetos:

- 1 dron
- 2 brazos (pertenecen al dron)
- 4 personas
- 5 localizaciones (de la cual una es el depósito)
- 7 cajas
- 2 tipos de contenido (medicina y comida)

Inicializamos los datos tal que:

- El dron empieza en el depósito y tiene dos brazos.
- Todas las cajas están en el depósito y contienen: Comida, medicina, comida, comida, medicina, medicina, comida (respectivamente a las 7 cajas).
- Las personas aparecen en las localizaciones, que dado nombres como: (persona1, persona 2, ..., persona4 y localización1, localización2, ..., localización4) están todos en las localizaciones coincidentes con su número y necesitan comida, medicina, comida, comida también respectivamente al número de la persona.

El objetivo del problema sería conseguir los suministros que piden las personas. Estos suministros son:

- persona1: comida
- persona2: medicina
- persona3: comida
- persona4: comida

2.4 Evaluación de Planificadores

Para evaluar los planificadores, hemos usado un tester de planificadores que realizaba soluciones a problemas generados aleatoriamente de dificultad creciente hasta que la búsqueda de una solución superase los 60 segundos.

El tester en cuestión, era un programa de Python y funcionaba de la siguiente manera:

FUNCIONES AUXILIARES:

time_limit(seconds): Iniciaba un contador de duración “seconds” que detenía la ejecución de la búsqueda de soluciones una vez superaba dicha cantidad.

find_newest_problem_file(directory=”src”): No usada. Buscaba el último archive cuyo nombre fuese (“drone_problem_*.pddl”), siendo el asterisco un sustituto donde podrían entrar cualquier combinación de caracteres.

generate_problem(drones, carriers, locations, presons, crates, goals): Llama al archive de generación de problemas que se ha mencionado anteriormente y le pasa exactamente los mismos parámetros que se le están pasando para que genere un problema. Una vez está el problema creado, devuelve la ruta. En caso de haber un error, no devuelve nada.

run_planner(domain_file, problem_file,planner_path, time_limit_seconds=60): Dado un dominio, un problema, un planificador y una cantidad determinada, hace que el planificador genere una solución del problema en base al dominio con un límite de 60 segundos. En caso de que no lo tenga, este devuelve la solución y el tiempo que le ha costado, y en caso de que no, devuelve un None, especificando que no se ha encontrado una solución, y el tiempo dado, como para señalizar que se ha excedido.

delete_problem_file(problem_file): Borra todos los archivos cuyo nombre sea “dron_problem_*.pddl”, donde el asterisco es un sustituto donde podrían entrar cualquier combinación de caracteres.

plot_results(sizes, times, solutions_found, max_size): Hace gráficos respecto a las soluciones encontradas de un problema. Este gráfico será posteriormente guardado como un png.

FUNCIÓN PRINCIPAL/MAIN:

Es una función que aplica problemas de coste incremental hasta que excede alguno de los límites establecidos.

Los parámetros intraducibles son:

- --planner: Se le introduce la ruta en la que está alojado un planificador para ponerlo a prueba
- --domain: Se le pasa la ruta del dominio del problema para que sepa el planificador que normas, predicados y tipos tiene el problema
- --start-size: Seguido de un número establece el tamaño inicial del problema.
- --max-size: Seguido de un número establece el tamaño máximo del problema. Uno de los límites que puede llegar a ser superado y limitar la continuación del programa.
- --timeout: Seguido de un número indica el tiempo máximo que puede tomar un planificador para solucionar un problema.
- --continue-on-fail: Continúa las pruebas, aunque haya fallos consecutivos.

Para ejecutar el archivo de test_planiffier.py, se ha facilitado un archivo .sh de nombre “ejecutar.sh” y de la forma:

```
python3 src/test_planiffiers.py \
    --planner 'planificadores/lpg-td' \
    --domain 'pddl/dominio-drones.pddl' \
    --start-size 2 \
    --max-size 100 \
    --timeout 60

# Funciona ff, sgplan40 Y lpg-td
```

Que mediante el cambio del planificador en la opción del --planner entre las opciones:

- **planificadores/ff**
- **planificadores/lpg-td**
- **planificadores/sgplan40**

realizarían la ejecución del tester de la planificación pasada por parámetro.

Mediante esta llamada, se declaran los parámetros que especifican:

- **El planificador, que será el pasado. En el mismo comando mostrado anteriormente sería el LPG-TD**
- **El dominio, que será el establecido para esta parte (pddl/dominio-drones.pddl)**
- **El tamaño inicial de los problemas de dificultad ascendente que se crearán (en el ejemplo de 2)**
- **El tamaño máximo de los problemas de dificultad ascendente que se crearán (en el ejemplo de 100)**
- **El tiempo máximo que le tomará al planificador encontrar una solución de los problemas creados.**

Los planificaciones que hemos usado son:

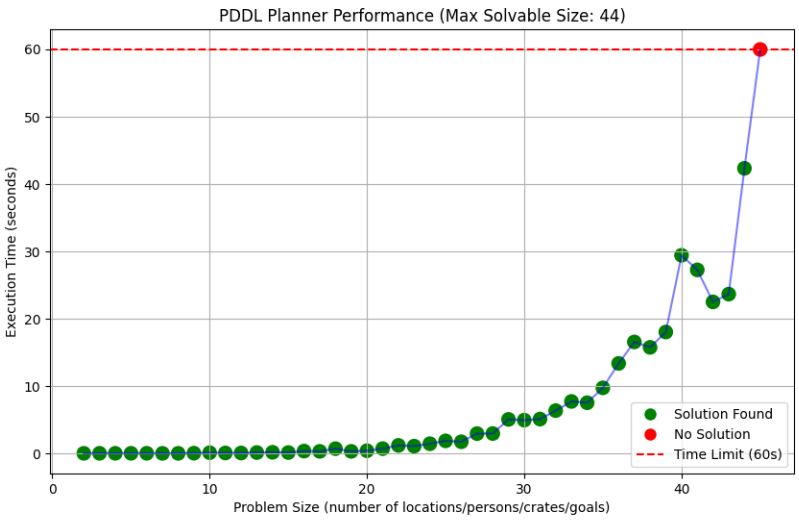
FF:

Cualidades:

- Generador de planes rápidamente sin optimizar en costos.
- Basado en heurísticas de relajación, donde prioriza buscar soluciones con pocas acciones, a acciones de bajo costo.
- Compatible con PDDL STRIPS
- No maneja costes ni duraciones. Por lo que es muy rápido para llegar a soluciones, pero están poco optimizadas

Limitaciones:

- Al no considerar costes ni duraciones de acciones, genera soluciones que pueden llegar a ser caros.
- Debido a su forma de aplicar la heurística, si no encuentra mejoras inmediatas, puede quedarse atascado.
- No soporta acciones durativas ni concurrencias, lo que lo hace ineficiente en dominios



Cómo podemos observar en la búsqueda de soluciones de problemas dados, encuentra el límite en problemas se tamaño 44.

Solución:

```
step    0: VOLAR DRON1 DEPOSITO LOC22
        1: VOLAR DRON1 LOC22 LOC4
        2: COGER DRON1 CAJA16 LOC4 BRAZO2
        3: COGER DRON1 CAJA10 LOC4 BRAZO1
        4: ENTREGAR DRON1 CAJA10 BRAZO1 PERS6 LOC4 MEDICINA
        5: VOLAR DRON1 LOC4 LOC24
        6: COGER DRON1 CAJA30 LOC24 BRAZO1
        7: ENTREGAR DRON1 CAJA16 BRAZO2 PERS5 LOC24 COMIDA
        8: VOLAR DRON1 LOC24 LOC16
        9: VOLAR DRON1 LOC16 LOC25
       10: COGER DRON1 CAJA9 LOC25 BRAZO2
       11: ENTREGAR DRON1 CAJA30 BRAZO1 PERS14 LOC25 MEDICINA
       12: COGER DRON1 CAJA13 LOC25 BRAZO1
       13: ENTREGAR DRON1 CAJA9 BRAZO2 PERS14 LOC25 COMIDA
       14: VOLAR DRON1 LOC25 LOC26
       15: VOLAR DRON1 LOC26 LOC9
       16: COGER DRON1 CAJA23 LOC9 BRAZO2
       17: ENTREGAR DRON1 CAJA13 BRAZO1 PERS23 LOC9 MEDICINA
       18: COGER DRON1 CAJA4 LOC9 BRAZO1
       19: ENTREGAR DRON1 CAJA23 BRAZO2 PERS26 LOC9 COMIDA
       20: VOLAR DRON1 LOC9 LOC26
       21: VOLAR DRON1 LOC26 LOC30
       22: COGER DRON1 CAJA1 LOC30 BRAZO2
       23: ENTREGAR DRON1 CAJA4 BRAZO1 PERS25 LOC30 MEDICINA
       24: COGER DRON1 CAJA7 LOC30 BRAZO1
       25: VOLAR DRON1 LOC30 LOC26
       26: ENTREGAR DRON1 CAJA7 BRAZO1 PERS16 LOC26 MEDICINA
       27: VOLAR DRON1 LOC26 LOC6
       28: COGER DRON1 CAJA28 LOC6 BRAZO1
       29: VOLAR DRON1 LOC6 LOC7
       30: ENTREGAR DRON1 CAJA28 BRAZO1 PERS1 LOC7 MEDICINA
       31: COGER DRON1 CAJA29 LOC7 BRAZO1
       32: ENTREGAR DRON1 CAJA29 BRAZO1 PERS24 LOC7 MEDICINA
       33: VOLAR DRON1 LOC7 LOC22
       34: COGER DRON1 CAJA3 LOC22 BRAZO1
       35: VOLAR DRON1 LOC22 LOC16
       36: VOLAR DRON1 LOC16 LOC23
       37: ENTREGAR DRON1 CAJA1 BRAZO2 PERS13 LOC23 COMIDA
       38: VOLAR DRON1 LOC23 LOC12
       39: COGER DRON1 CAJA2 LOC12 BRAZO2
       40: VOLAR DRON1 LOC12 LOC16
       41: ENTREGAR DRON1 CAJA2 BRAZO2 PERS2 LOC16 COMIDA
       42: VOLAR DRON1 LOC16 LOC14
       43: COGER DRON1 CAJA6 LOC14 BRAZO2
       44: VOLAR DRON1 LOC14 LOC16
       45: ENTREGAR DRON1 CAJA3 BRAZO1 PERS2 LOC16 MEDICINA
       46: VOLAR DRON1 LOC16 LOC29
       47: COGER DRON1 CAJA5 LOC29 BRAZO1
       48: VOLAR DRON1 LOC29 LOC27
       49: ENTREGAR DRON1 CAJA6 BRAZO2 PERS15 LOC27 COMIDA
       50: VOLAR DRON1 LOC27 LOC24
       51: COGER DRON1 CAJA8 LOC24 BRAZO2
       52: VOLAR DRON1 LOC24 LOC27
       53: ENTREGAR DRON1 CAJA5 BRAZO1 PERS15 LOC27 MEDICINA
       54: VOLAR DRON1 LOC27 LOC19
       55: COGER DRON1 CAJA14 LOC19 BRAZO1
       56: VOLAR DRON1 LOC19 LOC6
       57: ENTREGAR DRON1 CAJA8 BRAZO2 PERS17 LOC6 COMIDA
       58: VOLAR DRON1 LOC6 LOC15
       59: COGER DRON1 CAJA11 LOC15 BRAZO2
       60: VOLAR DRON1 LOC15 LOC6
       61: ENTREGAR DRON1 CAJA14 BRAZO1 PERS17 LOC6 MEDICINA
       62: VOLAR DRON1 LOC6 LOC25
       63: COGER DRON1 CAJA15 LOC25 BRAZO1
       64: VOLAR DRON1 LOC25 LOC1
       65: ENTREGAR DRON1 CAJA15 BRAZO1 PERS8 LOC1 MEDICINA
       66: VOLAR DRON1 LOC1 LOC25
       67: COGER DRON1 CAJA18 LOC25 BRAZO1
       68: VOLAR DRON1 LOC25 LOC1
       69: ENTREGAR DRON1 CAJA18 BRAZO1 PERS19 LOC1 MEDICINA
       70: VOLAR DRON1 LOC1 LOC28
       71: COGER DRON1 CAJA19 LOC28 BRAZO1
       72: VOLAR DRON1 LOC28 LOC30
       73: ENTREGAR DRON1 CAJA11 BRAZO2 PERS22 LOC30 COMIDA
       74: VOLAR DRON1 LOC30 LOC29
       75: COGER DRON1 CAJA12 LOC29 BRAZO2
       76: VOLAR DRON1 LOC29 LOC30
       77: ENTREGAR DRON1 CAJA12 BRAZO2 PERS25 LOC30 COMIDA
       78: VOLAR DRON1 LOC30 LOC5
       79: COGER DRON1 CAJA17 LOC5 BRAZO2
       80: VOLAR DRON1 LOC5 LOC30
       81: ENTREGAR DRON1 CAJA17 BRAZO2 PERS29 LOC30 COMIDA
       82: VOLAR DRON1 LOC30 LOC3
       83: COGER DRON1 CAJA22 LOC3 BRAZO2
       84: VOLAR DRON1 LOC3 LOC30
       85: VOLAR DRON1 LOC30 LOC11
       86: ENTREGAR DRON1 CAJA22 BRAZO2 PERS10 LOC11 COMIDA
       87: VOLAR DRON1 LOC11 LOC30
       88: ENTREGAR DRON1 CAJA19 BRAZO1 PERS29 LOC30 MEDICINA
       89: VOLAR DRON1 LOC30 LOC5
       90: COGER DRON1 CAJA20 LOC5 BRAZO2
       91: VOLAR DRON1 LOC5 LOC11
       92: ENTREGAR DRON1 CAJA20 BRAZO2 PERS10 LOC11 MEDICINA
       93: VOLAR DRON1 LOC11 LOC13
       94: COGER DRON1 CAJA21 LOC13 BRAZO2
       95: VOLAR DRON1 LOC13 LOC11
       96: ENTREGAR DRON1 CAJA21 BRAZO2 PERS30 LOC11 MEDICINA
       97: VOLAR DRON1 LOC11 DEPOSITO
```

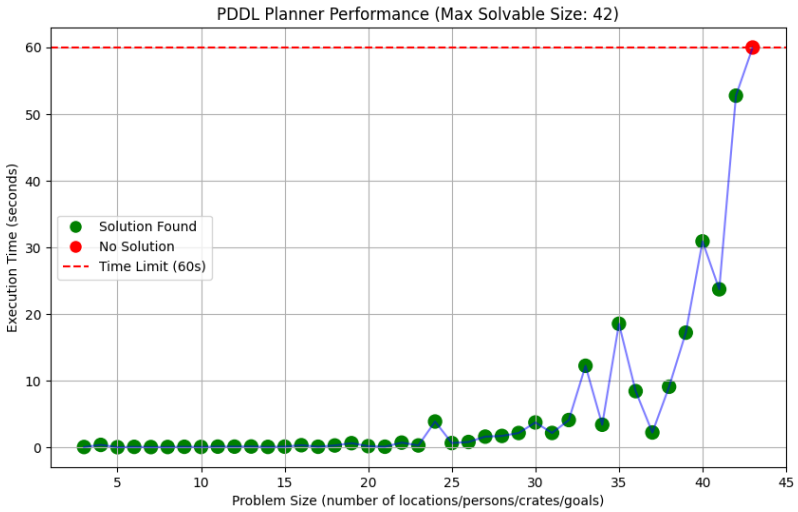
LPG-TD:

Cualidades:

- Planificador basado en búsqueda local con optimizaciones heurísticas
- Soporta costes y duraciones de acciones
- Genera múltiples planes en paralelo y permite optimización en distintos criterios
- Compatible con PDDL STRIPS
- Es supuestamente eficiente en problemas grandes y con restricciones de tiempo

Limitaciones:

- Puede ser algo lento en problemas sin restricciones temporales
- Basado en su búsqueda local, puede generar soluciones subóptimas
- Puede requerir ajustes manuales para mejorar soluciones en ciertos dominios



El gráfico nos muestra que es capaz de encontrar resultados hasta alcanzar un tamaño de 42.

Solución:

```
; Parsing time 0.00
; Mutex time 0.02
; NrActions 98

0:   (VOLAR DRON1 DEPOSITO LOC30) [1]
1:   (COGER DRON1 CAJA1 LOC30 BRAZO2) [1]
2:   (VOLAR DRON1 LOC30 LOC25) [1]
3:   (COGER DRON1 CAJA18 LOC25 BRAZO1) [1]
4:   (VOLAR DRON1 LOC25 LOC30) [1]
5:   (ENTREGAR DRON1 CAJA18 BRAZO1 PERS25 LOC30 MEDICINA) [1]
6:   (VOLAR DRON1 LOC30 LOC16) [1]
7:   (ENTREGAR DRON1 CAJA1 BRAZO2 PERS2 LOC16 COMIDA) [1]
8:   (VOLAR DRON1 LOC16 LOC24) [1]
9:   (COGER DRON1 CAJA8 LOC24 BRAZO1) [1]
9:   (COGER DRON1 CAJA30 LOC24 BRAZO2) [1]
10:   (ENTREGAR DRON1 CAJA8 BRAZO1 PERS5 LOC24 COMIDA) [1]
11:   (VOLAR DRON1 LOC24 LOC1) [1]
12:   (ENTREGAR DRON1 CAJA30 BRAZO2 PERS19 LOC1 MEDICINA) [1]
13:   (VOLAR DRON1 LOC1 LOC24) [1]
14:   (VOLAR DRON1 LOC24 LOC20) [1]
15:   (COGER DRON1 CAJA26 LOC20 BRAZO1) [1]
16:   (VOLAR DRON1 LOC20 LOC23) [1]
17:   (ENTREGAR DRON1 CAJA26 BRAZO1 PERS13 LOC23 COMIDA) [1]
18:   (VOLAR DRON1 LOC23 LOC3) [1]
19:   (COGER DRON1 CAJA22 LOC3 BRAZO2) [1]
20:   (VOLAR DRON1 LOC3 LOC11) [1]
21:   (ENTREGAR DRON1 CAJA22 BRAZO2 PERS10 LOC11 COMIDA) [1]
22:   (VOLAR DRON1 LOC11 LOC5) [1]
23:   (COGER DRON1 CAJA20 LOC5 BRAZO1) [1]
24:   (VOLAR DRON1 LOC5 LOC11) [1]
25:   (ENTREGAR DRON1 CAJA20 BRAZO1 PERS30 LOC11 MEDICINA) [1]
26:   (VOLAR DRON1 LOC11 LOC9) [1]
27:   (COGER DRON1 CAJA23 LOC9 BRAZO1) [1]
28:   (VOLAR DRON1 LOC9 LOC25) [1]
29:   (ENTREGAR DRON1 CAJA23 BRAZO1 PERS14 LOC25 COMIDA) [1]
29:   (COGER DRON1 CAJA13 LOC25 BRAZO2) [1]
30:   (COGER DRON1 CAJA9 LOC25 BRAZO1) [1]
31:   (VOLAR DRON1 LOC25 LOC1) [1]
32:   (ENTREGAR DRON1 CAJA13 BRAZO2 PERS8 LOC1 MEDICINA) [1]
33:   (VOLAR DRON1 LOC1 LOC25) [1]
34:   (COGER DRON1 CAJA15 LOC25 BRAZO2) [1]
35:   (ENTREGAR DRON1 CAJA15 BRAZO2 PERS14 LOC25 MEDICINA) [1]
36:   (VOLAR DRON1 LOC25 LOC27) [1]
37:   (ENTREGAR DRON1 CAJA9 BRAZO1 PERS15 LOC27 COMIDA) [1]
38:   (VOLAR DRON1 LOC27 LOC14) [1]
39:   (COGER DRON1 CAJA27 LOC14 BRAZO1) [1]
40:   (VOLAR DRON1 LOC14 LOC6) [1]
41:   (ENTREGAR DRON1 CAJA27 BRAZO1 PERS17 LOC6 COMIDA) [1]
41:   (COGER DRON1 CAJA28 LOC6 BRAZO2) [1]
42:   (VOLAR DRON1 LOC6 LOC9) [1]
43:   (ENTREGAR DRON1 CAJA28 BRAZO2 PERS23 LOC9 MEDICINA) [1]
44:   (VOLAR DRON1 LOC9 LOC6) [1]
45:   (VOLAR DRON1 LOC6 LOC5) [1]
46:   (COGER DRON1 CAJA17 LOC5 BRAZO2) [1]
47:   (VOLAR DRON1 LOC5 LOC30) [1]
48:   (ENTREGAR DRON1 CAJA17 BRAZO2 PERS22 LOC30 COMIDA) [1]
49:   (VOLAR DRON1 LOC30 LOC4) [1]
50:   (COGER DRON1 CAJA16 LOC4 BRAZO1) [1]
50:   (COGER DRON1 CAJA10 LOC4 BRAZO2) [1]
51:   (ENTREGAR DRON1 CAJA10 BRAZO2 PERS6 LOC4 MEDICINA) [1]
52:   (VOLAR DRON1 LOC4 LOC30) [1]
53:   (ENTREGAR DRON1 CAJA16 BRAZO1 PERS25 LOC30 COMIDA) [1]
54:   (VOLAR DRON1 LOC30 LOC29) [1]
55:   (COGER DRON1 CAJA5 LOC29 BRAZO2) [1]
56:   (VOLAR DRON1 LOC29 LOC30) [1]
57:   (ENTREGAR DRON1 CAJA5 BRAZO2 PERS29 LOC30 MEDICINA) [1]
58:   (VOLAR DRON1 LOC30 LOC29) [1]
59:   (COGER DRON1 CAJA12 LOC29 BRAZO2) [1]
60:   (VOLAR DRON1 LOC29 LOC9) [1]
61:   (ENTREGAR DRON1 CAJA12 BRAZO2 PERS26 LOC9 COMIDA) [1]
62:   (COGER DRON1 CAJA4 LOC9 BRAZO2) [1]
63:   (VOLAR DRON1 LOC9 LOC11) [1]
64:   (ENTREGAR DRON1 CAJA4 BRAZO2 PERS10 LOC11 MEDICINA) [1]
65:   (VOLAR DRON1 LOC11 LOC9) [1]
66:   (VOLAR DRON1 LOC9 LOC12) [1]
67:   (COGER DRON1 CAJA2 LOC12 BRAZO1) [1]
68:   (VOLAR DRON1 LOC12 LOC30) [1]
69:   (ENTREGAR DRON1 CAJA2 BRAZO1 PERS29 LOC30 COMIDA) [1]
69:   (COGER DRON1 CAJA7 LOC30 BRAZO2) [1]
70:   (VOLAR DRON1 LOC30 LOC13) [1]
71:   (COGER DRON1 CAJA21 LOC13 BRAZO1) [1]
72:   (VOLAR DRON1 LOC13 LOC26) [1]
73:   (ENTREGAR DRON1 CAJA21 BRAZO1 PERS16 LOC26 MEDICINA) [1]
74:   (VOLAR DRON1 LOC26 LOC13) [1]
75:   (VOLAR DRON1 LOC13 LOC27) [1]
76:   (ENTREGAR DRON1 CAJA7 BRAZO2 PERS15 LOC27 MEDICINA) [1]
77:   (VOLAR DRON1 LOC27 LOC7) [1]
78:   (COGER DRON1 CAJA29 LOC7 BRAZO2) [1]
79:   (ENTREGAR DRON1 CAJA29 BRAZO2 PERS1 LOC7 MEDICINA) [1]
80:   (VOLAR DRON1 LOC7 LOC20) [1]
81:   (COGER DRON1 CAJA25 LOC20 BRAZO2) [1]
82:   (VOLAR DRON1 LOC20 LOC7) [1]
83:   (ENTREGAR DRON1 CAJA25 BRAZO2 PERS24 LOC7 MEDICINA) [1]
84:   (VOLAR DRON1 LOC7 LOC28) [1]
85:   (COGER DRON1 CAJA19 LOC28 BRAZO1) [1]
86:   (VOLAR DRON1 LOC28 LOC6) [1]
87:   (ENTREGAR DRON1 CAJA19 BRAZO1 PERS17 LOC6 MEDICINA) [1]
88:   (VOLAR DRON1 LOC6 LOC22) [1]
89:   (COGER DRON1 CAJA3 LOC22 BRAZO1) [1]
90:   (VOLAR DRON1 LOC22 LOC16) [1]
91:   (ENTREGAR DRON1 CAJA3 BRAZO1 PERS2 LOC16 MEDICINA) [1]
92:   (VOLAR DRON1 LOC16 DEPOSITO) [1]
```

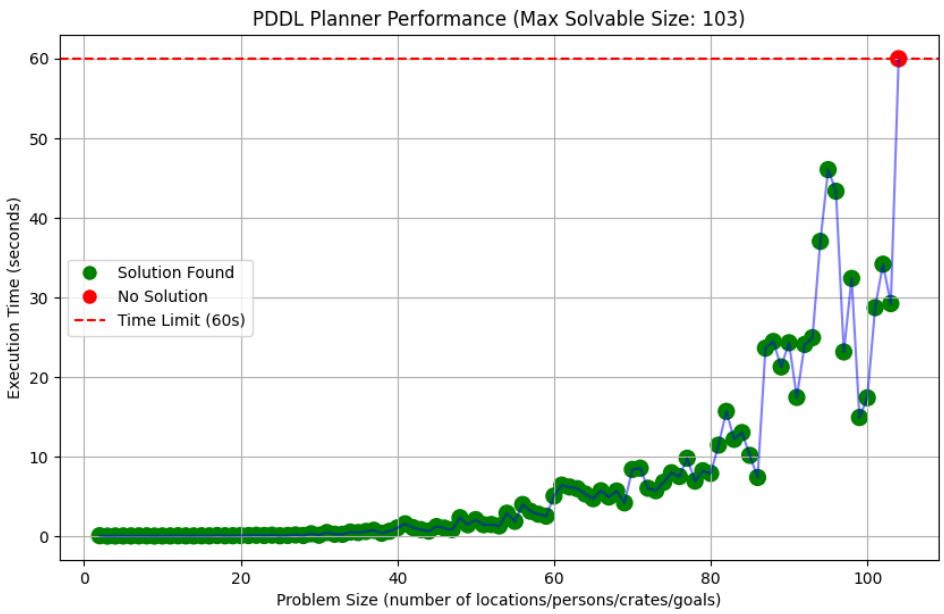
SGPLAN40:

Cualidades:

- Planificador basado descomposición de problemas en subproblemas más pequeños
- Soporta acciones durativas, concurrencia y restricciones temporales
- Compatible con PDDL STRIPS
- Eficiente en dominios con restricciones y planificación multi-objetivo.
- Maneja bien problemas de gran escala donde otros planificadores se pueden quedar atascados

Limitaciones:

- Su descomposición de problemas puede generar planes inconsistentes si los subprogramas no están bien coordinados
- No siempre encuentra la solución globalmente más óptima debido a su método de segmentación
- Puede ser sensible a la estructura del problema



Superando las capacidades de los dos planificadores anteriores, este modelo es capaz de solucionar problemas hasta de un tamaño de 104.

Solución:

```
; Parsing time 0.00
; Mutex time 0.02
; NrActions 98

0:   (VOLAR DRON1 DEPOSITO LOC30) [1]
1:   (COGER DRON1 CAJA1 LOC30 BRAZO2) [1]
2:   (VOLAR DRON1 LOC30 LOC25) [1]
3:   (COGER DRON1 CAJA18 LOC25 BRAZO1) [1]
4:   (VOLAR DRON1 LOC25 LOC30) [1]
5:   (ENTREGAR DRON1 CAJA18 BRAZO1 PERS25 LOC30 MEDICINA) [1]
6:   (VOLAR DRON1 LOC30 LOC16) [1]
7:   (ENTREGAR DRON1 CAJA1 BRAZO2 PERS2 LOC16 COMIDA) [1]
8:   (VOLAR DRON1 LOC16 LOC24) [1]
9:   (COGER DRON1 CAJA8 LOC24 BRAZO1) [1]
9:   (COGER DRON1 CAJA30 LOC24 BRAZO2) [1]
10:   (ENTREGAR DRON1 CAJA8 BRAZO1 PERS5 LOC24 COMIDA) [1]
11:   (VOLAR DRON1 LOC24 LOC1) [1]
12:   (ENTREGAR DRON1 CAJA30 BRAZO2 PERS19 LOC1 MEDICINA) [1]
13:   (VOLAR DRON1 LOC1 LOC24) [1]
14:   (VOLAR DRON1 LOC24 LOC20) [1]
15:   (COGER DRON1 CAJA26 LOC20 BRAZO1) [1]
16:   (VOLAR DRON1 LOC20 LOC23) [1]
17:   (ENTREGAR DRON1 CAJA26 BRAZO1 PERS13 LOC23 COMIDA) [1]
18:   (VOLAR DRON1 LOC23 LOC3) [1]
19:   (COGER DRON1 CAJA22 LOC3 BRAZO2) [1]
20:   (VOLAR DRON1 LOC3 LOC11) [1]
21:   (ENTREGAR DRON1 CAJA22 BRAZO2 PERS10 LOC11 COMIDA) [1]
22:   (VOLAR DRON1 LOC11 LOC5) [1]
23:   (COGER DRON1 CAJA20 LOC5 BRAZO1) [1]
24:   (VOLAR DRON1 LOC5 LOC11) [1]
25:   (ENTREGAR DRON1 CAJA20 BRAZO1 PERS30 LOC11 MEDICINA) [1]
26:   (VOLAR DRON1 LOC11 LOC9) [1]
27:   (COGER DRON1 CAJA23 LOC9 BRAZO1) [1]
28:   (VOLAR DRON1 LOC9 LOC25) [1]
29:   (ENTREGAR DRON1 CAJA23 BRAZO1 PERS14 LOC25 COMIDA) [1]
29:   (COGER DRON1 CAJA13 LOC25 BRAZO2) [1]
30:   (COGER DRON1 CAJA9 LOC25 BRAZO1) [1]
31:   (VOLAR DRON1 LOC25 LOC1) [1]
32:   (ENTREGAR DRON1 CAJA13 BRAZO2 PERS8 LOC1 MEDICINA) [1]
33:   (VOLAR DRON1 LOC1 LOC25) [1]
34:   (COGER DRON1 CAJA15 LOC25 BRAZO2) [1]
35:   (ENTREGAR DRON1 CAJA15 BRAZO2 PERS14 LOC25 MEDICINA) [1]
36:   (VOLAR DRON1 LOC25 LOC27) [1]
37:   (ENTREGAR DRON1 CAJA9 BRAZO1 PERS15 LOC27 COMIDA) [1]
38:   (VOLAR DRON1 LOC27 LOC14) [1]
39:   (COGER DRON1 CAJA27 LOC14 BRAZO1) [1]
40:   (VOLAR DRON1 LOC14 LOC6) [1]
41:   (ENTREGAR DRON1 CAJA27 BRAZO1 PERS17 LOC6 COMIDA) [1]
41:   (COGER DRON1 CAJA28 LOC6 BRAZO2) [1]
42:   (VOLAR DRON1 LOC6 LOC9) [1]
43:   (ENTREGAR DRON1 CAJA28 BRAZO2 PERS23 LOC9 MEDICINA) [1]
44:   (VOLAR DRON1 LOC9 LOC6) [1]
45:   (VOLAR DRON1 LOC6 LOC5) [1]
46:   (COGER DRON1 CAJA17 LOC5 BRAZO2) [1]
47:   (VOLAR DRON1 LOC5 LOC30) [1]
48:   (ENTREGAR DRON1 CAJA17 BRAZO2 PERS22 LOC30 COMIDA) [1]
49:   (VOLAR DRON1 LOC30 LOC4) [1]
50:   (COGER DRON1 CAJA16 LOC4 BRAZO1) [1]
50:   (COGER DRON1 CAJA10 LOC4 BRAZO2) [1]
51:   (ENTREGAR DRON1 CAJA10 BRAZO2 PERS6 LOC4 MEDICINA) [1]
52:   (VOLAR DRON1 LOC4 LOC30) [1]
53:   (ENTREGAR DRON1 CAJA16 BRAZO1 PERS25 LOC30 COMIDA) [1]
54:   (VOLAR DRON1 LOC30 LOC29) [1]
55:   (COGER DRON1 CAJA5 LOC29 BRAZO2) [1]
56:   (VOLAR DRON1 LOC29 LOC30) [1]
57:   (ENTREGAR DRON1 CAJA5 BRAZO2 PERS29 LOC30 MEDICINA) [1]
58:   (VOLAR DRON1 LOC30 LOC29) [1]
59:   (COGER DRON1 CAJA12 LOC29 BRAZO2) [1]
60:   (VOLAR DRON1 LOC29 LOC9) [1]
61:   (ENTREGAR DRON1 CAJA12 BRAZO2 PERS26 LOC9 COMIDA) [1]
62:   (COGER DRON1 CAJA4 LOC9 BRAZO2) [1]
63:   (VOLAR DRON1 LOC9 LOC11) [1]
64:   (ENTREGAR DRON1 CAJA4 BRAZO2 PERS10 LOC11 MEDICINA) [1]
65:   (VOLAR DRON1 LOC11 LOC9) [1]
66:   (VOLAR DRON1 LOC9 LOC12) [1]
67:   (COGER DRON1 CAJA2 LOC12 BRAZO1) [1]
68:   (VOLAR DRON1 LOC12 LOC30) [1]
69:   (ENTREGAR DRON1 CAJA2 BRAZO1 PERS29 LOC30 COMIDA) [1]
69:   (COGER DRON1 CAJA7 LOC30 BRAZO2) [1]
70:   (VOLAR DRON1 LOC30 LOC13) [1]
71:   (COGER DRON1 CAJA21 LOC13 BRAZO1) [1]
72:   (VOLAR DRON1 LOC13 LOC26) [1]
73:   (ENTREGAR DRON1 CAJA21 BRAZO1 PERS16 LOC26 MEDICINA) [1]
74:   (VOLAR DRON1 LOC26 LOC13) [1]
75:   (VOLAR DRON1 LOC13 LOC27) [1]
76:   (ENTREGAR DRON1 CAJA7 BRAZO2 PERS15 LOC27 MEDICINA) [1]
77:   (VOLAR DRON1 LOC27 LOC7) [1]
78:   (COGER DRON1 CAJA29 LOC7 BRAZO2) [1]
79:   (ENTREGAR DRON1 CAJA29 BRAZO2 PERS1 LOC7 MEDICINA) [1]
80:   (VOLAR DRON1 LOC7 LOC20) [1]
81:   (COGER DRON1 CAJA25 LOC20 BRAZO2) [1]
82:   (VOLAR DRON1 LOC20 LOC7) [1]
83:   (ENTREGAR DRON1 CAJA25 BRAZO2 PERS24 LOC7 MEDICINA) [1]
84:   (VOLAR DRON1 LOC7 LOC28) [1]
85:   (COGER DRON1 CAJA19 LOC28 BRAZO1) [1]
86:   (VOLAR DRON1 LOC28 LOC6) [1]
87:   (ENTREGAR DRON1 CAJA19 BRAZO1 PERS17 LOC6 MEDICINA) [1]
88:   (VOLAR DRON1 LOC6 LOC22) [1]
89:   (COGER DRON1 CAJA3 LOC22 BRAZO1) [1]
90:   (VOLAR DRON1 LOC22 LOC16) [1]
91:   (ENTREGAR DRON1 CAJA3 BRAZO1 PERS2 LOC16 MEDICINA) [1]
92:   (VOLAR DRON1 LOC16 DEPOSITO) [1]
```

Para poder comparar correctamente los distintos modelos, he seleccionado un problema de tamaño 30 para que lo resuelvan todos y lo he dejado en el apartado final de cada planificador con "solución".

- **Comparación de rendimiento**:

  La lectura del rendimiento de todos los planificadores, apuntan a que el mejor para generar una planificación en este contexto, es el SGPLAN-40, que es capaz de llegar a encontrar soluciones inferiores 60 segundos de tamaño de hasta 103.

  Además, podemos descartar de forma concluyente al planificador "ff", ya que no solo es por poco igual de lento que el LPG-TD, sino que tiene muchas menos funcionalidades y genera más pasos de los que generan los otros dos planificadores.

  Por otra parte, el LPG-TD, tiene la capacidad de buscar soluciones de forma simultánea, lo que puede impulsar su uso y de dichas soluciones, comparar para ver la óptima.

  Sin embargo, y volviendo al planificador inicial, es el más rápido, también optimiza la solución con el fin de encontrar una, aunque no siempre óptima, optimizada.

---

4. Conclusiones

Mediante el muestreo de resultados de planificaciones obtenidos, hemos demostrado la calidad de los planificadores con problemas de dificultad ascendente, mostrando la clara superioridad del SGPLAN-40, seguido de LPG-TD y finalmente el FF, donde además de mostrar una superioridad en la capacidad de resolución, han mostrado obtener resultados más optimizados.