PRÁCTICA 1

PARTE 2

Marcos Fúster Peña

Daniel de la Fuente Sánchez

Pablo Tirado Barragán

Memoria de la Práctica 1 – Planificación Clásica con PDDL

1. Introducción

1.1 Objetivo de la Práctica

El objetivo de esta práctica es ampliar los modelos de planificación usados en la primera parte de la práctica, aplicando las técnicas de planificación con tecnologías de manejos de costes de las acciones y por lo tanto optimizarla de cara a dichas acciones.

También es relevante comentar que ha habido cambios en el sistema, ya que ha aparecido un nuevo tipo en el dominio, los contenedores, que sirven para guardar un máximo de 4 cajas, haciendo más rentable los viajes.

Los objetivos específicos de la práctica incluyen:

- Cambio en el dominio para introducir los contenedores para que un dron sea capaz de transportar más cajas de un solo viaje y optimizarlo.
- Incorporar costes de acción.
- Implementar la representación de números mediante predicados en PDDL
- Evaluar el rendimiento de distintos planificadores con las modificaciones realizadas

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.

1.3 Contenido del Proyecto

El proyecto incluye los siguientes archivos y directorios:

PDDL

- parte2/pddl/dominio-drones-parte-2.pddl – Modelo del dominio de planificación.
- parte2/pddl/problema-parte-2.pddl – Instancia de problema generado como prueba

Generador de Problemas

- parte2/src/generate_problem.py – Script en Python para la generación de problemas.
- parte2/src/test_planiffier.py – Script en Python para generar gráficos de eficiencia de los planificadores dados.

Planificadores

- planificadores/fast-downward.sif
- planificadores/metricff

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

- **Modificaciones del dominio de la parte 1**:

El dominio de esta segunda parte añade costes de acciones y un nuevo tipo encargado de permitir el transporte de varias cajas simultáneamente.

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.
- 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

Y hemos añadido el nuevo tipo:

- contenedor: Transportador de cajas. Para su correcto funcionamiento, un dron debe estar libre para poder cargarlo.
- num: Identificador numérico que usaremos para aplicar la limitación del sistema
- Para añadir el contendor sin problemas, hemos eliminado los brazos, que de por si iban a ser muy incómodos de programar y verificar que no tengan nada cargado simultáneamente. No debería modificar la actuación del dron, ya que de por sí, un contendor es óptimo que los brazos.

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
- **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.

Y hemos aumentado algunos debido al nuevo tipo:

- **en-deposito** ?l - localización: Indica si una localización es un depósito
- **contenedor-en** ?k – contenedor ?l – localización: Indica la localización del contenedor
- **contenedor-libre** ?k - contenedor: Indica si un contendor no está siendo usado
- **dron-libre** ?d - dron: Indica si un dron no está cargando nada
- **tiene-contenedor** ?d – dron ?k – contenedor: Indica si un dron concreto tiene un contenedor concreto
- **tiene-caja** ?d -dron ?c -caja: Indica si un dron concreto tiene una caja concreta
- **caja-libre** ?c – caja: Indica si una caja no está siendo usada
- **siguiente** ?n1 – num ?n2 – num: Indica el siguiente número de una sucesión
- **cajas-en-contenedor** ?k - contenedor ?n – num: Indica cuantas cajas hay en un contenedor
- **cero** ?n – num: Indica si un número es 0

Funciones:

Ahora que se nos ha solicitado realizar cálculos como de distancias, cajas por contenedor y depósito de drones, es necesario añadir funciones, que son capaces de realizar cálculos y restricciones en base a su valor numérico.

Las usadas para este dominio son:

- **(fly-cost ?l1 - localizacion ?l2 - localizacion)**: Representa el costo en términos de recursos (combustible) de volar desde una localización (?l1) a otra (?l2).
- **(total-cost)**: Es un acumulador que mantiene el costo total del plan generado. Se usa para evaluar y minimizar los planes en términos de eficiencia.

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 recoge un contenedor del depósito

Precondiciones

- El dron (?d) debe estar en la misma localización (?l)
- La localización ?l debe ser un depósito
- El dron (?d) debe estar libre
- El contenedor (?k) debe estar libre (que no vacío)
- El contenedor (?k) debe estar en la localización (?l)
- El contenedor debe estar vacío (cero ?n)

Efectos

- El dron obtiene el contenedor (tiene-contenedor ?d ?k)
- El dron deja de estar libre (not (dron-libre ?d)).
- El contenedor deja de estar disponible (not (contenedor-libre ?k)).
- El contenedor ya no está en la localización (not (contenedor-en ?k ?l)).
- Se inicializa el número de cajas dentro del contenedor en cero (cajas-en-contenedor ?k ?n)

- ### Acción: Dejar

Descripción

El dron se desplaza de una localización a otra.

Precondiciones

- El dron (?d) debe estar en la localización (?l).
- La localización (?l) debe ser un depósito.
- El dron debe estar cargando un contenedor (?k).
- El contenedor debe estar vacío (cero ?n).
- El contenedor debe contener exactamente ?n cajas
- El dron debe tener suficiente combustible para el trayecto

Efectos

- El dron suelta el contenedor (not (tiene-contenedor ?d ?k)).
- El dron vuelve a estar libre (dron-libre ?d).
- El contenedor vuelve a estar en la localización (contenedor-en ?k ?l).
- El contenedor queda disponible para su uso (contenedor-libre ?k).

- ### Acción: coger-caja

Descripción

El dron recoge una caja de una localización

Precondiciones

- El dron (?d) debe estar en la misma localización (?l).
- La caja (?c) debe estar en la localización (?l).
- El dron no debe estar cargando otra caja (dron-sin-caja ?d).
- La caja debe estar libre (caja-libre ?c)

Efectos

- El dron obtiene la caja (tiene-caja ?d ?c).
- El dron ya no está sin carga (not (dron-sin-caja ?d)).
- La caja deja de estar libre (not (caja-libre ?c)).
- La caja desaparece de la localización (not (caja-en ?c ?l)).
-

Acción: meter

Descripción

El dron carga una caja centro de un contenedor.

Precondiciones

- El dron (?d) debe estar cargando un contenedor (?k).
- El dron debe estar cargando una caja (?c).
- Debe haber espacio disponible en el contenedor (siguiente ?n1 ?n2).
- El contenedor debe contener exactamente ?n1 cajas.

Efectos

- La caja pasa al interior del contenedor (en-contenedor ?k ?c).
- El dron deja de sostener la caja (not (tiene-caja ?d ?c)).
- El dron vuelve a estar sin carga (dron-sin-caja ?d).
- Se actualiza el número de cajas dentro del contenedor (not (cajas-en-contenedor ?k ?n1), cajas-en-contenedor ?k ?n2).

Acción: sacar

Descripción

El dron extrae una caja del contenedro.

Precondiciones

- El dron (?d) debe estar cargando un contenedor (?k).
- La caja (?c) debe estar dentro del contenedor (?k).
- El dron no debe estar cargando otra caja (dron-sin-caja ?d).
- Debe haber al menos una caja en el contenedor (siguiente ?n1 ?n2).

Efectos

- La caja es retirada del contenedor (not (en-contenedor ?k ?c)).
- Se actualiza el número de cajas dentro del contenedor (not (cajas-en-contenedor ?k ?n2), cajas-en-contenedor ?k ?n1).
- El dron obtiene la caja (tiene-caja ?d ?c).
- El dron ya no está sin carga (not (dron-sin-caja ?d)).

Acción: volar

Descripción

El dron se desplaza de una localización a otra.

Precondiciones

- El dron (?d) debe estar en la localización de origen (?from).
- El dron no debe estar cargando una caja (dron-sin-caja ?d).

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)).
- Se incrementa el costo total del plan en función del vuelo (increase (total-cost) (fly-cost ?from ?to)).

- ### Acción: entregar

Descripción

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

Precondiciones

- El dron (?d) debe estar en la misma localización (?l).
- El dron debe estar cargando una caja (?c).
- La persona (?p) debe estar en la misma localización (?l).
- La caja (?c) debe contener un contenido (?t).
- La persona (?p) debe necesitar ese contenido (necesita ?p ?t).

Efectos

- La persona (?p) obtiene el contenido (tiene ?p ?t).
- La persona deja de necesitar el contenido (not (necesita ?p ?t)).
- El dron deja de cargar la caja (not (tiene-caja ?d ?c)).
- El dron vuelve a estar sin carga (dron-sin-caja ?d).

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.

flight_cost(location_coords, location_num1, location_num2): Función proporcionada por el esqueleto. Calcula 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.

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).

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
- --contents: Seguido de un número, indica el número de contenedores que generará el programa (Actualizado) -> quita los brazos de la parte1
- --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.

Por supuesto, este archivo, aunque un esqueleto similar al de la parte 1, tiene diferencias fundamentales, como una de las mencionadas arriba, donde en lugar de usar “carriers”, los brazos del dron, lo cambiamos por “contents”, contenedores, la nueva herramienta de transporte de cajas.

También hemos cambiado el dominio al que hace referencia, llamando al nuevo dominio, dominio-drones-2.

EJEMPLO DE USO:

python generate_problem.py -d 1 -r 1 -l 5 -p 5 -c 5 -g 3 -o output_folder

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

```
(define (problem drone_problem_d1_l5_p5_c5_g3_ct2)
(:domain dominio-drones-2)

no commands
(:objects
  dron1 - dron
  deposito loc1 loc2 loc3 loc4 loc5 - localizacion
  caja1 caja2 caja3 caja4 caja5 - caja
  comida medicina - contenido
  pers1 pers2 pers3 pers4 pers5 - persona
  k1 - contenedor
  n0 n1 n2 n3 n4 - num
)

no commands
(:init
  (dron-en dron1 deposito)
  (dron-libre dron1)
  (dron-sin-caja dron1)

    (en-deposito deposito)
  (contenedor-en k1 deposito)
  (contenedor-libre k1)
  (cajas-en-contenedor k1 n0)
  (caja-en caja1 deposito)
  (caja-libre caja1)
  (contiene caja1 comida)
  (caja-en caja2 deposito)
  (caja-libre caja2)
  (contiene caja2 comida)
  (caja-en caja3 deposito)
  (caja-libre caja3)
  (contiene caja3 comida)

  (caja-en caja4 deposito)
  (caja-libre caja4)
  (contiene caja4 medicina)
  (caja-en caja5 deposito)
  (caja-libre caja5)
  (contiene caja5 medicina)

  (persona-en pers1 loc1)

  (persona-en pers2 loc2)

  (persona-en pers3 loc3)

  (persona-en pers4 loc4)
  (necesita pers4 comida)
  (necesita pers4 medicina)

  (persona-en pers5 loc5)
  (necesita pers5 comida)

  (= (fly-cost deposito deposito) 1)
  (= (fly-cost deposito loc1) 20)
  (= (fly-cost deposito loc2) 34)
  (= (fly-cost deposito loc3) 11)
  (= (fly-cost deposito loc4) 22)
  (= (fly-cost deposito loc5) 30)
  (= (fly-cost loc1 deposito) 20)
  (= (fly-cost loc1 loc1) 1)
  (= (fly-cost loc1 loc2) 26)
  (= (fly-cost loc1 loc3) 10)
  (= (fly-cost loc1 loc4) 9)
  (= (fly-cost loc1 loc5) 24)
```

```
    (= (fly-cost loc2 deposito) 34)
    (= (fly-cost loc2 loc1) 26)
    (= (fly-cost loc2 loc2) 1)
    (= (fly-cost loc2 loc3) 32)
    (= (fly-cost loc2 loc4) 17)
    (= (fly-cost loc2 loc5) 6)
    (= (fly-cost loc3 deposito) 11)
    (= (fly-cost loc3 loc1) 10)
    (= (fly-cost loc3 loc2) 32)
    (= (fly-cost loc3 loc3) 1)
    (= (fly-cost loc3 loc4) 16)
    (= (fly-cost loc3 loc5) 28)
    (= (fly-cost loc4 deposito) 22)
    (= (fly-cost loc4 loc1) 9)
    (= (fly-cost loc4 loc2) 17)
    (= (fly-cost loc4 loc3) 16)
    (= (fly-cost loc4 loc4) 1)
    (= (fly-cost loc4 loc5) 15)
    (= (fly-cost loc5 deposito) 30)
    (= (fly-cost loc5 loc1) 24)
    (= (fly-cost loc5 loc2) 6)
    (= (fly-cost loc5 loc3) 28)
    (= (fly-cost loc5 loc4) 15)
    (= (fly-cost loc5 loc5) 1)

    (cero n0)
    (siguiente n0 n1)
    (siguiente n1 n2)
    (siguiente n2 n3)
    (siguiente n3 n4)
    (= (total-cost) 0)
)

(:goal (and
    (dron-en dron1 deposito)
    (tiene pers4 comida)
    (tiene pers4 medicina)
    (tiene pers5 comida)
))

(:metric minimize (total-cost))
)
```

2.3 Ejemplo de problema:

El problema que vamos a tratar es el anterior generado:

Para entender el problema, vamos a desglosarlo y comentarlo:

En dicho problema, podemos identificar estos objetos:

- 1 dron
- 1 contenedores (inútiles porque solo usaremos 1, que es el máximo que puede llevar un dorn)
- 5 personas
- 5 localizaciones de las cuales una es el deposito
- 5 cajas
- 3 objetivos
- 2 tipos de contenido

Inicializamos los datos tal que:

- El dron empieza en el depósito y empieza libre
- Se declara el lugar y el valor que tiene que la localización llamada depósito para que sea donde se inician las cajas, contenedores y drones, además de ser el sitio donde se reposta.
- Todas las cajas están en el depósito y contienen: Comida, comida, comida, medicina y medicina (respectivamente a las 5 cajas).
- Las personas aparecen en las localizaciones, que dado nombres como: (persona1, persona 2, ..., persona5 y localización1, localización2, ..., localización5) están todos en las localizaciones coincidentes con su número y necesitan comida, medicina, comida, comida también respectivamente al número de la persona.
- Una vez creadas las zonas, se crea la distancia entre las zonas, especificándola mediante la función fly-cost.
- Se indican las sucesiones numéricas y el predicado 0 a unos de los especificados
- Se inicializa el coste total del viaje a 0 con la función total-cost

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

- persona1:
- persona2:
- persona3:
- perosna4: comida y medicina
- persona 5: 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, cost, solutions_found, max_size, planner=""):

(Actualizado) Hace gráficos respecto a las soluciones encontradas de un problema. Se pude especificar el tamaño del problema y los segundos máximos para estandarizar la gráfica, el coste del plan generado, las soluciones encontradas y el nombre que se le va a dar al archivo de salida. Este gráfico será posteriormente guardado como un png y de nombre se le pondrá el dado por el parámetro planner.

limpliar_carpeta_problemas(): (Nuevo) Elimina los archivos que ese encuentran dentro de la carpeta donde se guradan los problemas (parte2/problemasGenerados).

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.
- --step-size: Seguido de un número, establece el aumento de tamaño de problemas consecutivas
- --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 'seq-sat-fdss-2' \
    --domain 'pddl/dominio-drones-parte-2.pddl' \
    --step-size 5 \
    --start 2 \
    --max-size 1000 \
    --timeout 60

# downward: lama-first, seq-sat-fd-autotune-2, seq-sat-fdss-2, seq-opt-lmcut, seq-opt-bjolp, seq-opt-fdss-2
# planificadores/metricff
```

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

- **seq-sat-fdss-2**
- **lama-first**
- **seq-sat-fd-autotune-2**
- **seq-opt-lmcut**
- **seq-opt-bjolp**
- **seq-opt-fdss-2**
- **metricff**

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 seq-sat-fdss-2
- El dominio, que será el establecido para esta parte (pddl/dominio-drones-parte-2.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 aumento de dificultad de del problema en cierta cantidad. En el ejemplo 5, por lo que hay un margen de error en el encuentro del coste máximo de 5.
- El tiempo máximo que le tomará al planificador encontrar una solución de los problemas creados. (en el ejemplo 60 segundos)

Los planificaciones que hemos usado son:

SEQ-SAT-FDSS-2:

Aplicación:
Utiliza una búsqueda heurística con selección dinámica de estrategias, adaptándose según el problema.

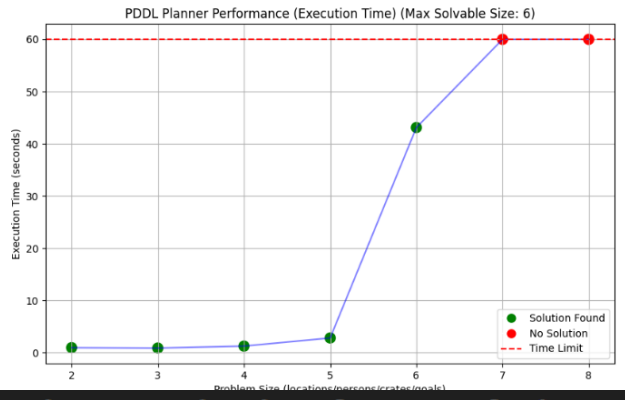
Cualidades:

- Planificador basado en Fast Downward con estrategias mixtas de búsqueda.
- Soporta heurística avanzada
- Equilibra la rapidez de generación con la calidad de solución
- No es necesario adaptarlo a dominios concretos, pues se adapta bien a distintos modelos
- Útil para los problemas con restricciones temporales, ya que busca resultados teniendo en cuenta el tiempo disponible.

Limitaciones:

- Soluciones subóptimas con problemas con acciones de alto costo
- Ya que satisface al usuario en cuanto a la búsqueda de soluciones con tiempo limitado, puede no ofrecer la solución óptima.

Grafico coste-tiempo:




```
ProblemSize,ExecutionTime,PlanCost,SolutionFound
2,0.9707,68.0,1
3,0.8969,148.0,1
4,1.2733,192.0,1
5,2.8257,380.0,1
6,43.1165,316.0,1
7,60.0000,N/A,0
8,60.0000,N/A,0
```

SEQ-SAT-FD-AUTOTUNE-2:

Aplicación:
Búsqueda heurística ajustando dinámicamente sus parámetros para optimizar eficiencia.

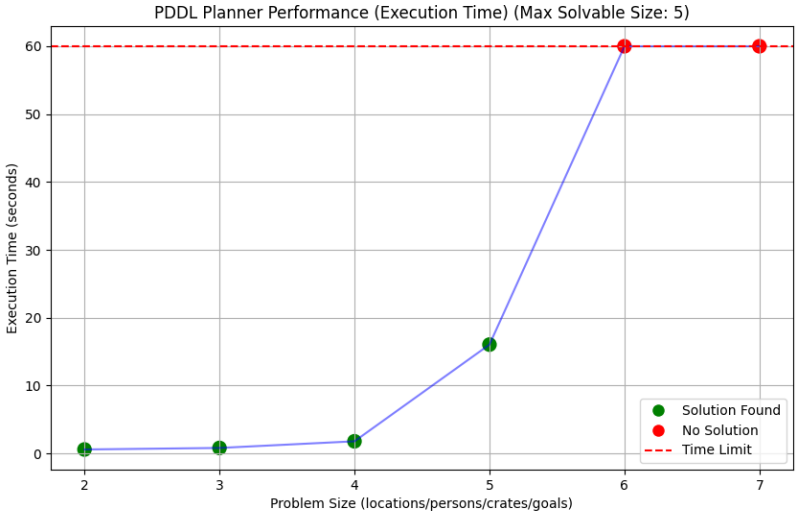
Cualidades:

- Variante de Fast Downward con ajuste de parámetros automatizado
- Ajusta la estrategia utiliza según el problema al que se enfrenta sin necesidad de aplicar una intervención manual.
- Debido a su gestión dependiente del problema, su adaptación tiene un coste temporal para generar soluciones respecto al modelo estándar, pero al adaptar el problema genera soluciones más optimizadas en un intervalo temporal más limitado.

Limitaciones:

- Puede ser ligeramente más lento que la versión estándar.
- Resultados subóptimos con restricciones temporales, ya que satisface el tiempo de búsqueda de soluciones solicitados por el usuario.

Gráfico coste – tiempo:




```
ProblemSize,ExecutionTime,PlanCost,SolutionFound
2,0.5908,38.0,1
3,0.8174,94.0,1
4,1.7967,188.0,1
5,16.0443,184.0,1
6,60.0000,N/A,0
7,60.0000,N/A,0
```

SEQ-OPT-BOJLP:

Aplicación:
Búsqueda en anchura con poda. Evalúa opciones óptimas y descarta soluciones innecesarias

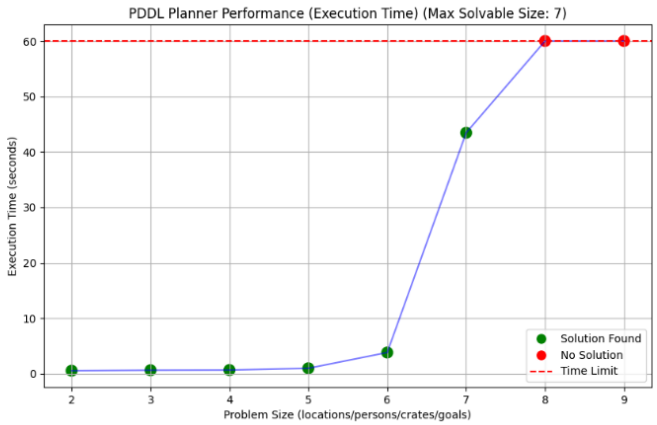
Cualidades:

- Planificador basado en programación lineal entera
- En problemas estructurados correctamente y con rutas de decisión de coste disparejo, genera soluciones óptimas
- Eficiente en problemas donde la segmentación lineal es efectiva, es decir, que la división en problemas más pequeños no afecta en gran medida a la solución de la suma de dichos problemas
- Sus soluciones siempre son óptimas

Limitaciones:

- En problemas de gran escala, extremadanamente lento (búsqueda en anchura con poda)
- Rendimiento dependiente de la formulación del problema, pudiendo no encontrar soluciones dependientes tanto por el tiempo o memoria necesaria.

Grafico coste tiempo




```
ProblemSize,ExecutionTime,PlanCost,SolutionFound
2,0.5520,73.0,1
3,0.6415,75.0,1
4,0.6718,63.0,1
5,0.9900,70.0,1
6,3.8310,109.0,1
7,43.4498,110.0,1
8,60.0000,N/A,0
9,60.0000,N/A,0
```

SEQ-OPT-IMCUT:

Aplicación:
Búsqueda en profundidad con heurística LM-cut: Localiza landmarks y corta acciones.

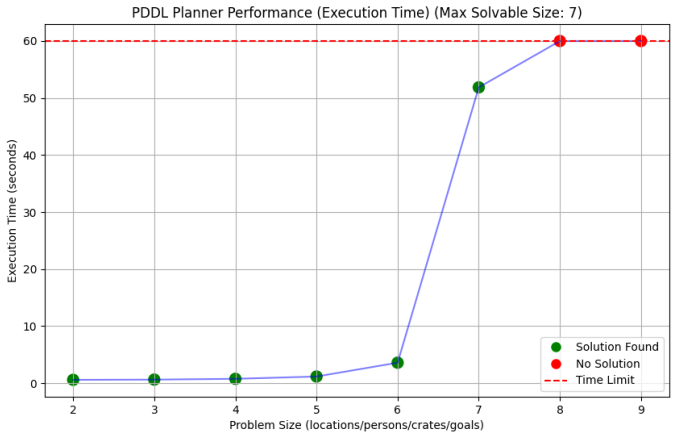
Cualidades:

- Planificador basado en la heurística LM-cut, especializado en soluciones óptimas
- Prioritario en dominios donde la heurística guía bien la búsqueda (en nuestro caso tiene sus variantes, ya que tiene facilidad si no hay rutas opcionales, así que depende de los valores opcionales. Si hay varios caminos de costo similar, puede desmejorar su aplicación)
- Siempre encuentra una solución óptima

Limitaciones:

- Lento en problemas grandes
- Cómo garantiza optimalidad, puede costarle encontrar la solución si hay varias soluciones.

Grafica coste-tiempo:




```
ProblemSize,ExecutionTime,PlanCost,SolutionFound
2,0.5697,68.0,1
3,0.6103,79.0,1
4,0.7448,101.0,1
5,1.1488,110.0,1
6,3.5719,57.0,1
7,51.8718,90.0,1
8,60.0000,N/A,0
9,60.0000,N/A,0
```

SEQ-OPT-FDSS-2:

Aplicación:
Búsqueda heurística con combinación de técnicas como LM-cut, A con múltiples heurísticas, poda de estados dominados y optimización basada en configuración automática de técnicas pare encontrar soluciones óptimas.

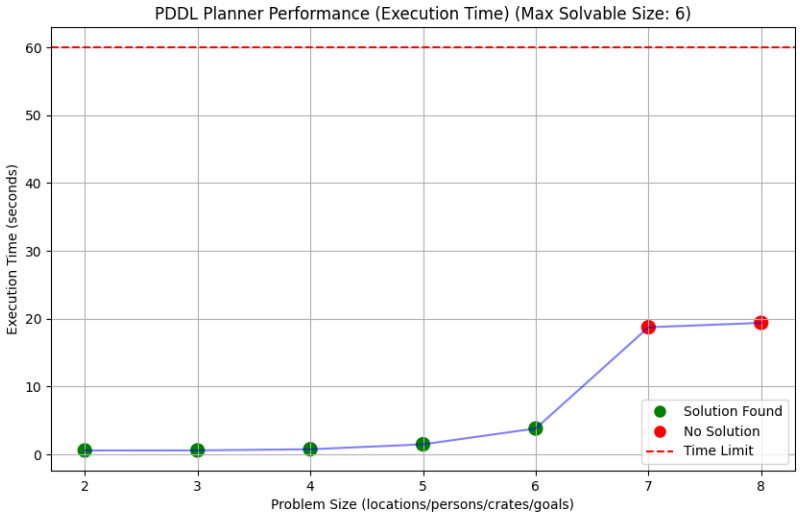
Cualidades:

- Variante de Fast Downward enfocada en optimalidad, siempre que encuentra una solución, es óptima.
- Usa múltiples técnicas para obtener soluciones más precisas y rápidas.

Limitaciones:

- Debido a la aplicación de distintas técnicas y decisión de cual tomar, el proceso es más lento que aplicar alguna concreta

Gráfico coste-tiempo:




```
ProblemSize,ExecutionTime,PlanCost,SolutionFound
2,0.5731,22.0,1
3,0.5883,69.0,1
4,0.7528,102.0,1
5,1.4814,107.0,1
6,3.8137,86.0,1
7,18.7358,N/A,0
8,19.3853,N/A,0
```

METRICFF:

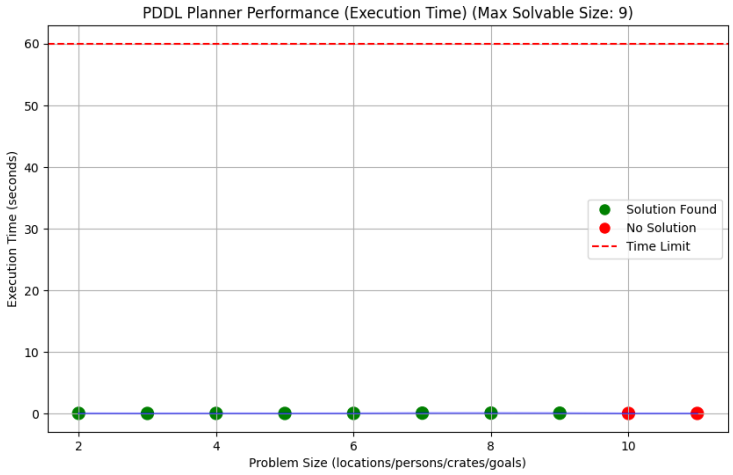
Aplicación:
Búsqueda heurística basada en FF adaptada para optimización de costos. Busca la acción más barata y realizable en cada momento.

Cualidades:

- Extensión de FF con soporte de métricas y costos
- Muy rápida en cuanto al encuentro de soluciones

Limitaciones:

- Debido a su baja optimización, el costo de la solución puede ser muy alto
- Debido a sus bajas capacidades de previsión, puede llegar a generar una solución muy poco óptima. Y parece ser, que según las pruebas, también es capaz de quedarse atascado




```
ProblemSize,ExecutionTime,PlanCost,SolutionFound
2,0.0175,8.0,1
3,0.0052,153.0,1
4,0.0131,258.0,1
5,0.0063,418.0,1
6,0.0153,458.0,1
7,0.0541,336.0,1
8,0.0597,684.0,1
9,0.0475,687.0,1
10,0.0044,N/A,0
11,0.0068,N/A,0
```

Lama-first:

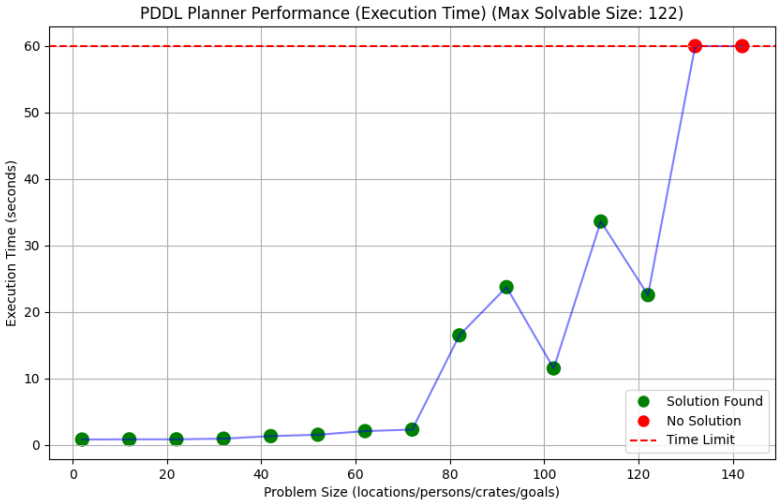
Aplicación:
Emplea una búsqueda heurística con relajación de costos, es decir, iterativa reiteradas veces, comprueba si ha mejorado el plan inicial y en tal caso, lo guarda y descarta el otro.

Cualidades:

- Extensión de FF con soporte de métricas y costos

Limitaciones:

- Debido a su baja optimización, el costo de la solución puede ser muy alto
- Debido a sus bajas capacidades de previsión, puede llegar a generar una solución muy poco óptima.




```
ProblemSize,ExecutionTime,PlanCost,SolutionFound
2,0.8027,80.0,1
12,0.8290,664.0,1
22,0.8298,1196.0,1
32,0.9357,1578.0,1
42,1.3149,2296.0,1
52,1.5247,2848.0,1
62,2.0647,3430.0,1
72,2.2942,4120.0,1
82,16.4752,4422.0,1
92,23.7351,4848.0,1
102,11.5017,5666.0,1
112,33.6101,5964.0,1
122,22.5355,6720.0,1
132,60.0000,N/A,0
142,60.0000,N/A,0
```

Para poder comparar correctamente los distintos modelos de planificación, debemos especificar el ámbito de uso que estamos aplicando. En el caso de transporte de mercancías en situaciones de emergencia, implica no solo rapidez en la búsqueda de una solución, sino que también requiere una solución rápida.

Comparación de rendimiento:

De forma teórica:

- **Planificadores descartados**:

    - **SEQ-OPT-BOJLP** y **SEQ-OPT-IMCUT** generaron soluciones óptimas en **todos los casos**, pero su **tiempo de ejecución suele ser significativamente mayor**, especialmente en problemas grandes. Estos modelos serían extremadamente útil en casos donde los solicitantes del servicio sean mínimos, pero debido a no poder asegurar esos parámetros, es mejor descartar su selección.

    - **SEQ-OPT-FDSS-2** implica una gran precisión en la búsqueda de soluciones optimizadas, siendo una muy viable para modelos que requieren de ahorro de costos por causas mayores, a lo que además, al aplicar distintas estrategias para buscar una solución, es ligeramente más rápida. Sin embargo, debido a su inestabilidad para problemas de grandes dimensiones, su búsqueda de soluciones limita en gran medida su efectividad.

- **Planificadores viables**:
    - **METRICFF** suele destacar en la generación de planes rápidos sin necesidad de ajustes manuales. Su capacidad de generar soluciones de forma rápida, hace que sea un modelo de gran valor para sistemas de emergencia, aunque debido a su baja optimización de los resultados encontrados, puede suponer una pérdida de tiempo. No sufre tanto en problemas de grandes dimensiones.
    - También, **LAMA-FIRST** ha demostrado la obtención de buenos resultados, adaptando, tardando ligeramente más en la búsqueda de soluciones, pero ofreciendo soluciones de calidad superior a las anteriores. Esto, aunque por poco y frente a selecciones ligeramente más optimizadas que el **METRICFF**, lo convierte en un modelo más adaptado para este tipo de casos.

- **<u>Planificadores óptimos</u>**:
  - Finalmente, **SEQ-SAT-FD-AUTOTUNE-2** tiene buenos resultados generalmente tanto en calidad como en rapidez de generación de solución, adaptando automáticamente sus estrategias para mejorar la eficiencia, aunque con un ligero costo adicional en tiempo debido a la necesidad de seleccionar un modelo y adaptarlo de forma manual, proveyendo mejores resultados, con más costo de planificación.
  - **SEQ-SAT-FDSS-2** destacó en la generación de planes rápidos sin necesidad de ajustes manuales. Su capacidad de **equilibrar rapidez y calidad de solución** lo hace ideal para **entornos de emergencia**, donde la respuesta rápida es prioritaria sobre la optimalidad absoluta. Resultando en el mejor planificador imaginable para este tipo de casos.

De forma práctica:
Podemos asignar las siguientes cualidades:

- metricFF no proporciona ni previsión ni optimización, además parece quedarse atascado en problemas de baja complejidad.
- Seq-sat-fd-autotune-2 parece quedarse estancado en problemas de complejidad muy baja. Entiendo que esto ocurre porque no ha recibido ningún tipo de apoyo manual, que hace que funcione mejor basados en distintos problemas, que es cuando obtiene su máximo potencial. Además, al no depender de encontrar la versión óptima de la solución, debería devolver un resultado viable.
- Seq-sat-fdss entiendo que tiene problemas antes que el resto, deteniéndose en problemas de complejidad 6 por tener que tomar la decisión de técnica, sin embargo, al igual que el seq-sat-fd-autotune-2, debería, en búsquedas más amplias, conseguir resultados, aunque no tan buenos, como los óptimos.
- Seq-opt-fdss-2 proporciona resultados impresionantes, de coste muy reducido, pero la cantidad de problemas que resuelve se empieza a disparar en problemas de complejidad 6. Cuanto más ascienda, más tardará en encontrar soluciones, por lo que para casos reducidos es muy útil, pero impracticables en complejidad alta.
- Seq-opt-imcut también tiene cierta utilidad al tener en cuestiones bajas del rango de 60 segundos como los usados, obteniendo resultados de con problemas de complejidad hasta 7. Sin embargo, se sabe que en rangos más altos, obtener resultados óptimos tendrían un tiempo de obtención mucho más altos.
- Lama-first es capaz de conseguir resultados siempre hasta problemas de complejidad 120 aproximadamente. Sin embargo, debido a su forma de mejorar soluciones iterativas, cambiándola cuando la solución nueva si es mejor. Esto, debido a la limitación temporal, genera soluciones muy poco optimizadas
- Seq-opt-bjolp también ofrece resultados optimizados, por lo que su acceso a soluciones en complejidades altas se limita mucho, tendiendo poco escalado. Es capaz de resolver problemas de complejidad 7.

4. Conclusiones

Para la resolución de problemas sobre servicios de transporte de mercancías de emergencia y por la necesidad de ofrecer inmediatez en sus servicios, se ha comprobado que la selección de planificadores más adaptada al mismo, son en orden y de menor a mayor conveniencia (y con matices de cantidad de servicios proveídos en una misma planificación):

- **SEQ-OPT-BOJLP** y **SEQ-OPT-IMCUT:** Lento para encontrar soluciones
- **SEQ-OPT-FDSS:** Lento para encontrar soluciones, pero adatando su estrategia de selección.
- **METRICFF:** Muy rápido en la búsqueda de soluciones. Muy poco optimizado
- **LAMA-FIRST:** Rápido en la búsqueda de soluciones. Mejora del problema en un tiempo estimado. A pesar de mejorar solución inicial, optimización vaga y baja.
- **SEQ-SAT-FD-AUTOTUNE-2:** Variedad en la velocidad de búsqueda de soluciones, optimizando la solución en base al tiempo permitido para encontrar la solución. Necesidad de adaptación manual para sacar todo su potencial, menos adaptación al problema.
- **SEQ-SAT-FDSS-2:** Variedad en la velocidad de búsqueda de soluciones, optimizando la solución en base al tiempo permitido para encontrar soluciones. Adaptación automática al problema.